MULTIMODAL TRAJECTORIES PREDICTION FOR AUTONOMOUS VEHICLES WITH LANE AWARENESS

BEIMING LI, JIARONG QI, YUELIN DANG

ABSTRACT. Trajectory prediction has been studied by numerous researchers for its significant applications in self-driving vehicles. The inherent uncertainty and the complexity of traffic scenes pose a barrier to accuracy. In traffic scenes, the target agent (the vehicle whose future trajectories we predict) interacts with other agents (other vehicles on the road) and is capable of a wide range of actions (turning, accelerating, braking). Our project aims to explore the use of ConvLSTM and attention layers in predicting vehicle trajectories given map information and past trajectories with lane awareness.

1. INTRODUCTION

Trajectory prediction refers to the prediction of a vehicle's future trajectories on the road given traffic context information. Accurate predictions of the target vehicle's future trajectories are essential for safe trajectory planning that efficiently achieves maneuvers and prevents collisions. The challenge for a solution model consists of two parts: firstly, the model must efficiently extract useful context information that correlates with or affects the target vehicle's trajectory planning. Secondly, the model must take uncertainties of the target vehicle's and its nearby vehicles' behaviors into account. The model must be robust to cope with such uncertainties and suggest multiple possible future trajectories. Researchers have studied this field comprehensively, and a large proportion of reliable solutions are based on deep neural networks [6].

We implemented Kim et al.'s approach of a lane-aware trajectory prediction model that utilizes target agent's past trajectories, lane coordinates, and past trajectories of one agent per lane that could potentially interact with the target agent [8]. We believe that the LaPred model overcomes the challenges of context information extraction and robustness against uncertainties. Not only do we implement the LaPred model, we also further Kim et al.'s research through multiple contributions:

- We implement a network variant that considers multiple nearby agents, or the multi-agent method, during the feature extraction stage, while the original paper considers a single nearby agent only.
- We justify the necessity of Modality Selection Block by implementing a network variant that ranks the likelihood of each modality naively and computes the modality selection loss with cosine similarity.
- We provide a modularized codebase, which makes model extension, ablation study, and comparison between different losses more convenient. The implementation is accessible at `https://github.com/beimingli0626/extended-LaPred`.

2. BACKGROUND AND RELATED WORK

In order to minimize risk of self-driving vehicles and promote safe autonomous systems, researchers have investigated trajectory prediction comprehensively. Trajectory prediction methods divide into subcategories of interaction-aware prediction and scene context-aware prediction [8]. Interaction-aware prediction studies the interaction of agents through grid-based and average pooling. Within this category, inputs are often taken as graphical structures. A remarkable example is Ivanovic and Pavone's approach to encode trajectory history and neighbor influence in a graph representation. They compare the performance of multiple LSTM methods and SGAN that predict future trajectories of human agents [5]. Another renowned research is Deo et al.'s multimodal approach on lane graphs encoded from HD maps [4].

On the other hand, scene context-aware prediction relies on scene context information extracted from spatial information and semantic maps. Phan-Minh et al. developed CoverNet to predict automobile trajectories, taking as input the current and past trajectories of all agents and HD semantic maps [10]. Also with scene context information, Sadeghian et al. use a different approach of a GAN-based model that observes physical constraints and social norms of human agents [11]. Other inspiring research in this field includes Salzmann et al.'s Trajectron++, a multi-agent behavior prediction model that synthesizes heterogeneous data to predict trajectories of multiple agents of multiple types, and Chandra et al.'s TraPHic, another multi-agent behavior prediction model that uses heterogeneous information to predict human, vehicle and animal trajectories in dense traffic scenario [12] [2].

We propose to use a scene context-aware prediction method that utilizes lane information, which is crucial to understanding the target agent's intentions. In a road setting, vehicles usually tend to follow a lane, change a lane, or make a turn. We propose that predicting the lane that the target vehicle intends to reach or keep and taking into account information of other agents on each lane that could interact with the target agent could lead to great model performance.

3. Approach

3.1. **Network Architecture.**

Our implementation is based on the original LaPred model proposed in [8] [7], which is illustrated in Figure 1.

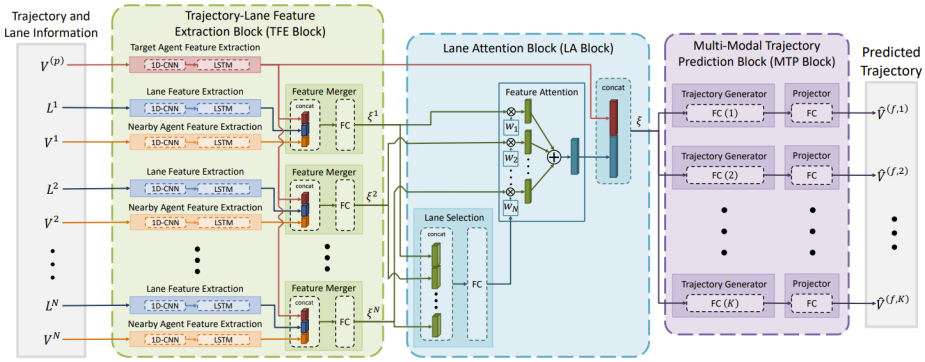

FIGURE 1. Original LaPred Architecture [8]

The entire architecture consists of three main sections: trajectory-lane feature extraction (TFE) block, lane-attention (LA) block, and multi-modal trajectory prediction (MTP) module. The following descriptions will focus mainly on our modifications and extensions to the original architecture.

3.1.1. *Trajectory-lane Feature Extraction.*

The TFE block presented in the original paper takes as inputs the target agent's past trajectories $V^{(p)}$, lane coordinates L^n, and the trajectories of one nearby agent selected per lane V^n. This information goes through lane feature extraction blocks, composed of CNN and LSTM, and outputs joint trajectory-lane features. However, our implementation provides another variant to processing inputs to the TFE block. Instead of inputting past trajectories of one nearby agent per lane per time, we concatenated past trajectories of both the selected nearby agent and agents on its adjacent lanes. For example, when processing the inputs of the nearby agent on the second lane V^2, we concatenated past trajectories of the nearby agent and the agents on its left and right lanes, thus concatenating V^1, V^2 and V^3. This is our multi-agent training mode. We denote the original mode as the single-agent mode.

We maintain the CNN-LSTM structure for feature extraction as such structure has been proven effective in encoding past trajectories of vehicles [13] [9]. LSTM is effective in processing and forecasting time series data in general. We could interpret the last hidden state of the LSTMs as sufficient statistics learned by the model. Then, we take the values of the last hidden state of the LSTMs as extracted feature representations and enter the lane-attention block.

3.1.2. *Lane-attention.*

The LA block weights and concatenates features extracted from the TFE block. The attention layers consist of fully connected layers and ReLU layers, and they transform feature inputs into valid probabilities for each lane for each sample in the batch. The per-lane features are then weighted by the probabilities and summed over all lanes to generate inputs ξ for the MTP module. A formula for the attention block could be expressed as follows:

$$w_i = softmax(B(ReLU(A\xi^i)) \,, \quad \xi = \sum_{i=1}^{N} w_i \xi^i$$

where N denotes the number of lanes, A and B are two sets of fully connected layers, ξ^1 to ξ^N are feature representations of trajectories and coordinates information per lane, and w_i represents how much attention we pay to lane feature ξ^i.

3.1.3. *Multimodal Trajectory Prediction and Selection.*

Given encoded environmental features ξ, we would like to learn trajectory distribution $p(V^{(f)}|\xi)$, where $V^{(f)}$ denotes future trajectories of target agent. Our implementation is a truncated version of the original MTP module, which is proposed in [3].

The first part of the original MTP module is Trajectory Predictions Block (TPBlock), which is shown in figure 1. TPBlock is composed of M trajectory generators, where each generator is composed of independent fully-connected

(FC) layers, followed by FC layers shared across generators. Generators take the concatenation of target agent features and LA-merged lane features as inputs. And we interpret the output of each generator as one possible future trajectory.

It is worth mentioning that our implementation truncates the shared FC layers. In [3], the author claims that the gradient only propagates through one generator among all M generators because of the specific loss function they choose. Hence, they would like to use shared FC layers to alleviate the potential over-fitting problem, as each generator is likely to be only trained by part of the entire dataset. However, we believe that the sub-dataset accessible to each generator is already large and diverse enough. Given our constrained computing power, the truncation of shared FC layers shouldn't hurt performance so much, while reducing a large amount of training time.

The second part is Modality Selection Block (MSBlock), which is composed of several FC layers with skip connections (not shown in figure 1). We concatenate each predicted trajectory with past trajectory of the target vehicle, to get the complete trajectory profile for each modality. Those profiles are then fed into MSBlock, which predicts the likelihood of each trajectory $p(V_i^f)$, such that $\sum_{i=1}^{M} p(V_i^f) = 1$. The hope is that MSBlock could learn about vehicle dynamics, and produce likelihood based on which trajectory is more dynamically feasible. Although we couldn't show what statistics are learned exactly, our experiment (section 4.4) shows that MSBlock is necessary for getting a good score on evaluation metrics (section 4.1).

3.2. **Loss.**

Our training loss is a weighted combination of lane selection loss, modality selection loss, and prediction loss.

Lane Selection Loss: Lane selection loss is the cross-entropy loss between model-predicted lane probabilities and the ground truth lane, which is the closest lane to the target agent's future true trajectory. It thus enables the model to pay more attention to the ground truth lane. We make it possible to weight the cross-entropy loss according to the distance between the ground truth lane and the predicted lane [8]. Namely, if the ground truth lane is far away from the predicted lane, this loss is given more weight in the sum of losses. This is because we would like to give more consideration to edge cases, where the predicted lane is starkly distant from the ground truth lane. This could happen in the case where a vehicle suddenly switches to a distant lane. Failure to predict such outcomes might lead to serious safety issues on the road. The weighting strategy could be expressed as

$$L_{\text{laneSelect}} = \frac{1}{b} \sum_{i=1}^{b} CE(\hat{\text{prob}}_i, \text{gt}_i) \times \text{ind}_i$$

where b denotes the batch size, $\hat{\text{prob}}_i$ denotes a vector of probabilities for each of the N lanes for the i^{th} prediction. Note that $\hat{\text{prob}}_i$ is the vector of w_i before softmax in Section 3.1.2. gt_i denotes the ground truth for the i^{th} data. ind_i is an integer value from 1 to 10 that represents the distance between the ground truth lane and the predicted lane. $\text{ind}_i = 1$ if the ground truth lane and the predicted lane are the same, and $\text{ind}_i = 10$ if the ground truth lane is the farthest from the predicted lane.

Modality Selection Loss (MSLoss): our TPBlock predicts multi-modality trajectories for one target vehicle. In real-world scenarios, it is valuable to rank modalities by their likelihood, so that autonomous vehicles could react to the most likely future of nearby agents. In order to investigate the role of MSBlock, we designed two different MSLosses.

In the first scenario, we include the MSBlock in the MTP module, which outputs a $\mathbb{R}^M$ vector, and each dimension represents the likelihood of one modality $p(V_i^f)$. We then rank the distance between modalities and ground truth future trajectory (GT), thus getting the index of modality closest to GT. The MSLoss is calculated with cross-entropy loss, which takes as inputs the $\mathbb{R}^M$ likelihood vector, and the index of closest modality as a one-hot vector.

In the second scenario, MSBlock is truncated, and we rank the modality naively. We implicitly assume that the first generator always generates the most likely trajectory, and the last generator always outputs the least likely one, therefore the ranking is simply **range**(M) in python. Then we calculate the distance between naive ranking and ground truth ranking with cosine similarity, which is widely used for comparing two rankings in literature. The performance under two scenarios is compared and discussed in section 4.4.

Prediction Losses: it is composed of position loss, which is the L1 loss between the best modality and ground truth trajectory, and lane-off loss, which reflects the tendency of the target agent to stick close to the reference lane in the future. The specific calculations are similar to Kim et al.'s implementation [8], therefore we skip the details here.

4. Experimental Results

4.1. **Evaluation Metrics.**

To evaluate our model performance, we establish the same evaluation metrics as the nuScenes prediction challenge: ADE (average displacement error) and FDE (final displacement error) as defined below. $d_i = ||s_i - s_i^{gt}||_2$ denotes the L2 distance between the predicted trajectory point and the ground truth trajectory point at timestamp i. $\{s_1, s_2, ..., s_t\}$

and $\{s_1^{gt}, s_2^{gt}, ..., s_t^{gt}\}$ denote the predicted and ground truth trajectory respectively. ADE thus calculates the average d across all trajectory points, while FDE calculates the final distance at timestamp t.

$$ADE = \frac{1}{t}\sum_{i=1}^{t} d_i \ , \quad FDE = d_t$$

Specifically, we evaluate the two metrics using different numbers (1, 5, and 10) of modalities: ADE1, ADE5, ADE10, FDE1, FDE5, FDE10. For example, for ADE5, we calculate the distance between the ground truth and the predicted trajectories at each moment element-wise, for five possible predicted trajectories. When selecting the five trajectories used for comparison, we rank the trajectories from the most probable to the least probable and take the first five. We then average the point-wise distance across time, and pick the smallest value among the five averages.

For FDE, instead of calculating distance point-wise along the trajectory, we consider only the final displacement. For example, for FDE5, we calculate only the distance between the ground truth's final point and the final displacement of the predicted trajectories, for the top five probable trajectories. Instead of averaging across time, we simply take the minimum of the five distance values.

We achieve state-of-the-art performance (refer to nuScenes leaderboard [1]) of 1.675 ADE5 and 1.365 ADE10 on validation data with one of our model variants (stepLR single-agent model that use weighted lane selection loss).

4.2. **Model Performance Comparison: Single-agent Models and Multi-agent Models.**

We find that the number of nearby agents whose past trajectories are used during feature extraction does not seem to affect model performance. As shown in Figure 2, the evaluation metrics for two model variants have similar trend and value. Even during training, where the discrepancies between models tend to show, single-agent models have approximately the same performance as that of multi-agent models for both ADE and FDE metrics.

This is probably because the way we concatenate trajectories for multi-agent models interferes with the time-sequence LSTMs. For example, suppose the time sequence has length T, when we consider the nearby agent and its two adjacent agents, we concatenate them sequentially such that in our concatenated sequence of length $3T$, the first T elements refer to the coordinates of the left adjacent agent, the $T+1$ to $2T$ elements the coordinates of the nearby agent, and the last T elements the coordinates of the right adjacent agent. The concatenated sequence of length $3T$, therefore, does not follow a strict time-sequence order. Thus, this preprocessing technique might not demonstrate LSTM's advantage in dealing with time-sequence data.

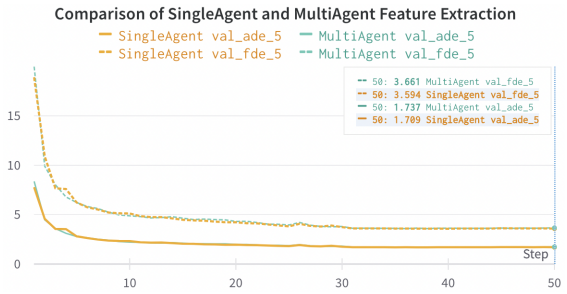

FIGURE 2. Single-agent vs. Multi-agent

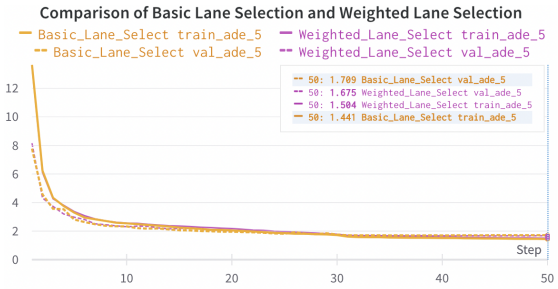

FIGURE 3. Basic vs. Weighted Lane Select

4.3. **Model Performance Comparison: Weighted Loss.**

We find that the inclusion of weighting for lane selection loss does not boost model performance. As shown in Figure 3, not only is the weighting method fruitless, the ADE5 metric for training is about 40 centimeters higher for the model with weighting, and about 5 centimeters higher for the validation process.

The method of weighting the lane selection loss is adopted in order to give more weight to edge cases such as the vehicle suddenly switching several lanes. We would need to give consideration to edge cases like this for safety reasons. However, it does not help boost training and validation results, probably for the below reasons:

- The proportion of edge cases is lower than expected. Most people drive gently following traffic rules, therefore the need to give more emphasis to edge cases was not as urgent as we expected.
- It could also be that the weighting method works effectively, and the consideration for edge cases is necessary, but since the edge cases account for only a small portion of the entire dataset, even if the model makes better predictions on those cases, it is not sufficient to boost model performance. Besides, to predict better results for edge cases, the parameters must be adjusted in their favor, thus compromising slightly model performance on common cases.

4.4. **Model Performance Comparison: With & Without Modality Selection Block.**

We would like to investigate how Modality Selection Block boosts our performance. We train the network on two different modality selection strategies as described in section 3.2: 1) use MSBlock to learn modality likelihood and compute cross-entropy loss; 2) naively assign a ranking to each modality, and compute the distance to ground truth ranking with cosine similarity (denoted as Without_MSBlock in figure).

As shown in Figure 4, we notice that both strategies achieve similar Validation ADE10, which indicates that they both learn a full set of trajectories that is similarly good on average. However, MSBlock achieves lower ADE5 (1.709) while the naive strategy doesn't learn well. The large gap in ADE5 indicates that MSBlock could correctly assign likelihood based on the trajectory profile of each modality.

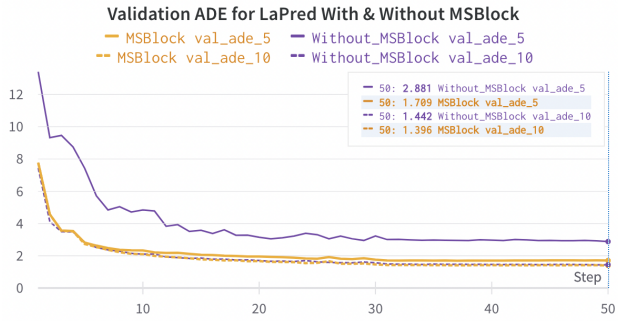

FIGURE 4. With vs. Without MSBlock

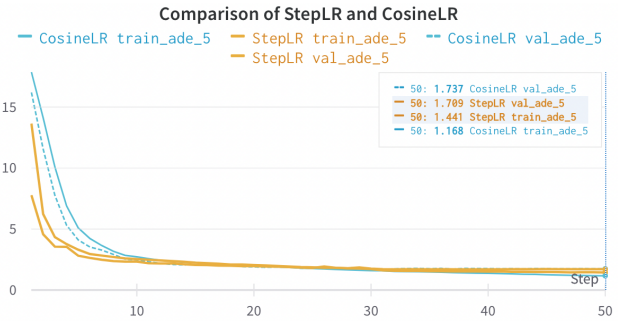

FIGURE 5. StepLR vs. CosineLR

4.5. **Model Performance Comparison: Schedulers.**

We also compare two types of learning rate schedulers: CosLR and StepLR. If StepLR (using Pytorch standard implementation) is used, the learning rate starts at 0.001, and is multiplied with 0.1 every 30 epochs. If CosLR is selected, we use the cosine learning rate schedule outlined in homework 4.

Through experimental results as shown in Figure 5, we discover that models with StepLR learn significantly faster in the first few epochs because of the higher starting learning rate. However, during validation, the performance of the model with StepLR is approximately the same as that with CosLR when the number of epochs increases. During training, models with CosLR seem to have lower metric values than those with StepLR, but since the difference does not show in validation, this indicates that models with CosLR tend to overfit, while they do not have better validation performances.

5. DISCUSSION

In this project, we implemented a multi-model trajectory prediction architecture based on previous research, with our customized extensions. Our best model variants could achieve 1.675 ADE5 and 1.365 ADE10 on validation data, which is close to state-of-the-art approaches.

Not only does our model achieve satisfying validation results, we also compare and contrast the model performance of single-agent models and multi-agent methods, models with and without weighting for lane selection loss, models with and without modality selection block, and models with various schedulers. If given sufficient time and resources, we would like to explore in-depth another way of incorporating multiple adjacent agents' trajectories into the feature extraction process.

Our existing method of multi-agent models does not bring about better model performance. We give one possible explanation in Section 4.2. We enrich nearby agent's past trajectory information by concatenating it with adjacent agents' past trajectories, and the concatenated sequence is not a strict time sequence. To lower validation metrics, we may also consider other ways to add multi-agent information besides direct concatenation. For example, concatenating the LSTM-encoded features of multiple agents instead, which we believe to be a good walk-around for the time sequence problem but require much larger GPU memory.

We could also explore a graph representation of lane information. Our current representation of lane information consists of lane segments and agents' trajectories represented in Cartesian coordinates, but a graph representation could potentially include more information such as the relationship and interactions between the lanes and their agents [4].

REFERENCES

[1] nuscenes challenge leaderboard. https://www.nuscenes.org/prediction?externalData=all/&mapData=all/&modalities=Any. Accessed: 2022-12-17.

[2] Rohan Chandra, Uttaran Bhattacharya, Aniket Bera, and Dinesh Manocha. Traphic: Trajectory prediction in dense and heterogeneous traffic using weighted interactions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8483–8492, 2019.

[3] Henggang Cui, Vladan Radosavljevic, Fang-Chieh Chou, Tsung-Han Lin, Thi Nguyen, Tzu-Kuo Huang, Jeff Schneider, and Nemanja Djuric. Multimodal trajectory predictions for autonomous driving using deep convolutional networks. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 2090–2096, 2019.

[4] Nachiket Deo, Eric Wolff, and Oscar Beijbom. Multimodal trajectory prediction conditioned on lane-graph traversals. In *Conference on Robot Learning*, pages 203–212. PMLR, 2022.

[5] Boris Ivanovic and Marco Pavone. The trajectron: Probabilistic multi-agent trajectory modeling with dynamic spatiotemporal graphs. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2375–2384, 2019.

[6] Phillip Karle, Maximilian Geisslinger, Johannes Betz, and Markus Lienkamp. Scenario understanding and motion prediction for autonomous vehicles-review and comparison. *IEEE Transactions on Intelligent Transportation Systems*, 2022.

[7] ByeoungDo Kim, Seong Hyeon Park, Seokhwan Lee, Elbek Khoshimjonov, Dongsuk Kum, Junsoo Kim, Jeong Soo Kim, and Jun Won Choi. Lapred github repository by kim et al. https://github.com/bdokim/LaPred. Accessed: 2022-12-17.

[8] ByeoungDo Kim, Seong Hyeon Park, Seokhwan Lee, Elbek Khoshimjonov, Dongsuk Kum, Junsoo Kim, Jeong Soo Kim, and Jun Won Choi. Lapred: Lane-aware prediction of multi-modal future trajectories of dynamic agents. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14636–14645, 2021.

[9] Seong Hyeon Park, ByeongDo Kim, Chang Mook Kang, Chung Choo Chung, and Jun Won Choi. Sequence-to-sequence prediction of vehicle trajectory via lstm encoder-decoder architecture. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1672–1678. IEEE, 2018.

[10] Tung Phan-Minh, Elena Corina Grigore, Freddy A. Boulton, Oscar Beijbom, and Eric M. Wolff. Covernet: Multimodal behavior prediction using trajectory sets. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14062–14071, 2019.

[11] Amir Sadeghian, Vineet Kosaraju, Ali Sadeghian, Noriaki Hirose, Hamid Rezatofighi, and Silvio Savarese. Sophie: an attentive gan for predicting paths compliant to social and physical constraints. In Abhinav Gupta, Derek Hoiem, Gang Hua, and Zhuowen Tu, editors, *Proceedings - 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2019*, pages 1349–1358, United States of America, 2019. IEEE, Institute of Electrical and Electronics Engineers.

[12] Tim Salzmann, Boris Ivanovic, Punarjay Chakravarty, and Marco Pavone. Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data. In *European Conference on Computer Vision*, pages 683–700. Springer, 2020.

[13] Guo Xie, Anqi Shangguan, Rong Fei, Wenjiang Ji, Weigang Ma, and Xinhong Hei. Motion trajectory prediction based on a cnn-lstm sequential model. *Science China Information Sciences*, 63(11):1–21, 2020.