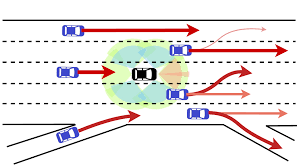
https://medium.com/swlh/review-multimodal-trajectory-predictions-for-autonomous-driving-using-deep-convolutional-networks-596edf8bc102

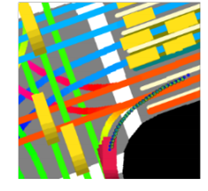
**Review: Multimodal Trajectory Predictions for Autonomous Driving Using Deep Convolutional Networks**

In this blog post, we will review the paper “Multimodal Trajectory Predictions for Autonomous Driving using Deep Convolutional Networks” by Henggang Cui et al. But before we start observing the paper itself, let’s check what is trajectory prediction at the first place?



In the real world, there are a lot of various situations, where one wants to predict where the actor (it can be anything — a person, a car, a plane, no matter what) will move next step (step also can be different — it can be millisecond, second or even a minute). But how to specify what is the “prediction” itself? What information do we need?

In simple language, the main goal of trajectory prediction is to determine a set of coordinates (it can be any-dimensional) that are connected to a specific point of time. So for a predicted actor, one can choose a time frame and then check which position it has. But here is the question — why do we need this information and why is it specifically important for autonomous driving?

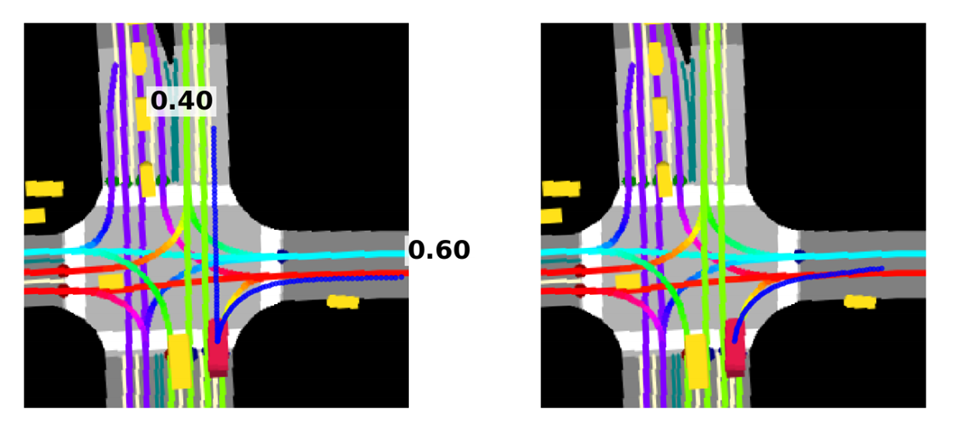


First of all, there are plenty of problems in self-driving vehicles: there is a high uncertainty of traffic behavior and a large number of different situations. That means that one cannot use a discrete number of situations and a discrete number of car movements. Second, the main tasks of self-driving vehicles, like ensuring safe and efficient operations and anticipating a multitude of possible behaviors of traffic actors in its surroundings, provide a large need in knowing the position of all vehicles beforehand.

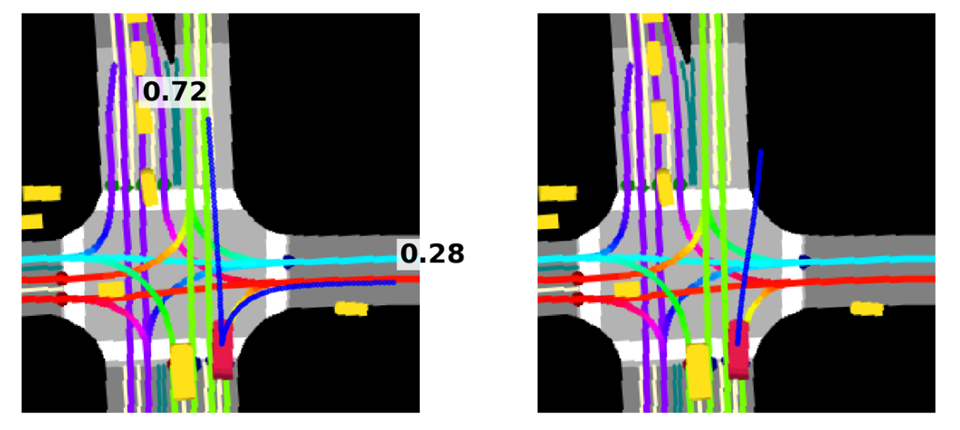
And here is where “multimodal” comes to action: what is that in case of trajectory prediction? Multimodal means that we have multiple predictions for each timeframe. Moreover, we will have a probability of each prediction — simplifying, 60% that vehicle will go forward and 40% that it will go right.

The main point of having this multimodality is to remove too generalized solutions. A lot of different algorithms have this problem — with quite similar inputs, it assumes, that it is the same situation, while in reality, it is not. It can be seen in this example.

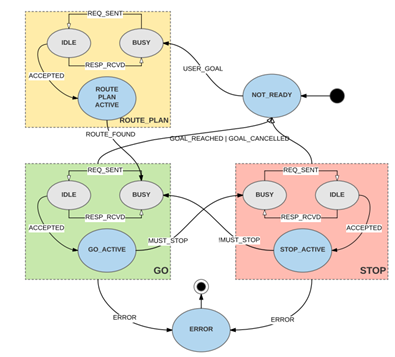
While the car is pretty far away from the turn, both multimodal (left) and unimodal (right) approaches predict the expected path to be turning to the right. For multimodal, we will, apparently, check the option having the maximum probability.



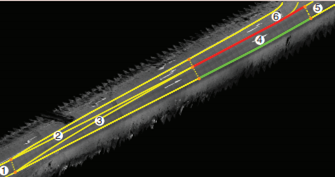
But when the vehicle moves a little bit forward, everything changes — the unimodal algorithm is starting to think “I thought, that it will go to the right, but it is moving forward right now. So I think *THE TRUTH IS OUT THERE —* somewhere in the middle between movement and predicted behavior”. At the same time, multimodal one doesn’t change its answer — it simply recalculates probabilities, and now the most possible option is moving forward and less possible one is turning to the right.



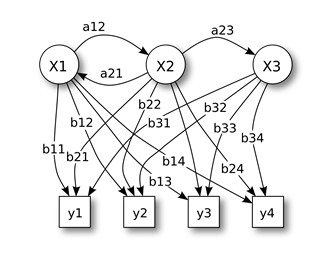
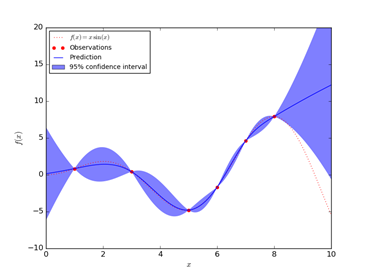
Now we understand why we want this multimodality and why we need trajectory prediction at all. Let’s check what was done by other people before. One of the ideas was to propagate the actor’s state over time based on assumptions of the underlying physical system. It had its own positive and negative sides — this approach is really good for short-term predictions because of understanding how the actor “works” itself, but is bad for long-term predictions as it ignores the surrounding context.



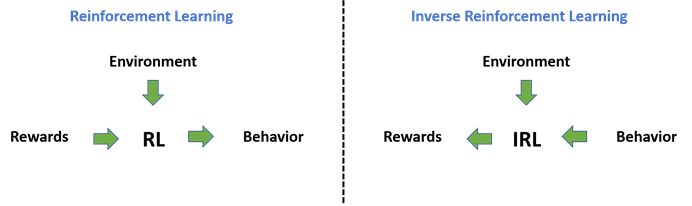
The other idea was to use map analysis — it is the method that as the first step associates each detected vehicle with one or more lanes from the map and then generates all possible paths for each “vehicle”-“lane” pair based on map topology, lane connectivity, and vehicle’s current state estimate. This method, in contrary to previous, is quite good for long-term predictions and tends to make reasonable predictions in common cases but is really sensitive to errors in the association of vehicles and lanes. Moreover, it cannot predict the “strange” behavior of an actor.



Now, let’s switch to machine learning methods and approaches. There are very effective methods based on Hidden Markov Models, Bayesian Networks and Gaussian processes, but unfortunately, they were not applicable for data that was collected by the authors of this paper and for their experience.



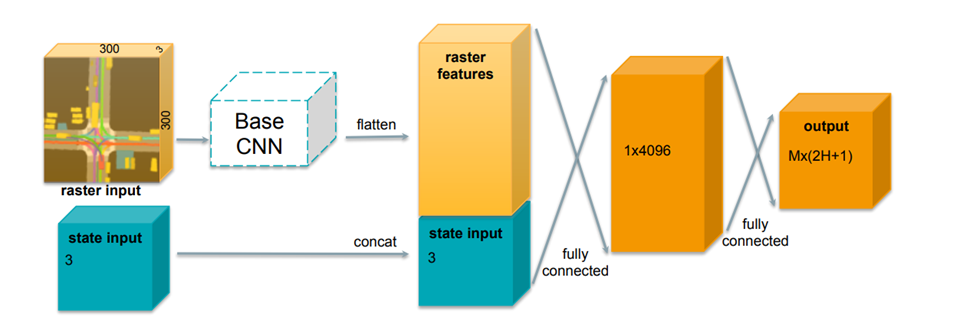
The other idea was to use Inverse Reinforcement Learning by modeling the environmental context. There inverse optimal control was used to predict pedestrian paths by considering scene semantics. However, the existing IRL methods are inefficient for real-time applications like the author’s idea.



RNN, including LSTM and GRU, also work quite well for paths and trajectories prediction. Though the authors of this paper assumed, that it was not possible to use this approach to build potential multimodality of trajectories. Authors also considered Mixture Density Networks as a potential working approach, but they were difficult to train due to numerical instabilities when operating in high-dimensional space.

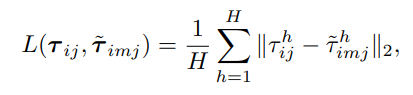
Authors made a decision to use CNN for this work, considering their data (more about it later). We will check the architecture a little bit later, but now let’s switch to the possible “multimodality” handling. There are few approaches like the ensemble of networks, but the authors also checked other options. First was teaching a model that assigns probabilities to N maneuver classes.Unfortunately, this approach requires a predefined discrete set of possible maneuvers, which may be hard to define for complex city driving. The other idea was to generate multimodal predictions through sampling, but it required repeated forward passes to generate multiple trajectories. In the end, the authors of the paper decided to use the “closest prediction” method. That means that they would use a single network producing M different outputs, each of them representing different hypotheses. In the end, they would use a loss that only accounts for the closest prediction to ground truth labels.

We’ve talked a lot about the authors’ data, so let’s start to go through it. First of all, each actor has state S, which includes bounding box, position, velocity, acceleration, heading and heading changing rate. Second, we have discrete times T for each event. In the end, we have a high definition map, which includes operation area, road locations, crosswalk locations, and lane directions. Having all this info, authors generate an actor-specific BEV raster image encoding the actor’s map surrounding and neighboring actors, which they used in the network shown below.



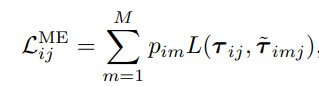
First, this map goes through the base CNN network. MobileNet-v2 was used for this approach, while for a unimodal task from the authors’ previous paper there were better options. I cannot say, why this one was chosen, but it is only one of the upcoming questions to this paper. Then the result was flattened and concatenated with state input, which included velocity, acceleration and heading change rate of an actor. In the end, it went through the fully connected layer and received a result which included M modes, for each mode, there was a probability of it, and H timestamps with 2 coordinates each. That gives us M\*(2\*H+1) output for one network.

But how can we get the right answer with the output? We need to think about the loss function first. Let’s define *single-mode loss,* which will be used later:



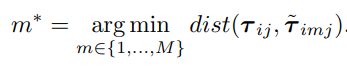
Here, **i** — is actor’s number, **m** — mode’s number, **t** — is time. So here we have a displacement (or 2-norm) error between ground truth trajectory for actor **i** at time **t** in the mode **m,** averaged by the horizon (the scope — some next moments of time) of this trajectory. We defined this loss to try to use it in more complicated losses, that we will try later.

Let’s start with straightforward Mixture-of-Experts loss.

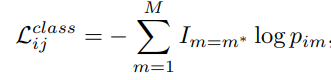


It uses the *single-mode loss,* that we defined before as L, and also it uses the probability of each model. It seems that it can be a valid solution, but in reality, it is not suitable for the trajectory prediction problem due to the mode collapse problem — all solutions look the same.

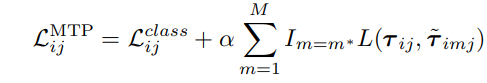
The other idea is *Multi-Trajectory Prediction Loss*. Let’s define it step by step:



We first run the forward pass of the neural network to obtain M output trajectories. We then identify mode **m∗** that is closest to the ground-truth trajectory according to an arbitrary trajectory distance function. This distance function is optional (displacement — using coordinates, or angular — using angle), but the main idea is to find the **best** mode of the proposed ones.



Then, using this **m\*** mode and binary indicator function **I**, we find *class loss*, which helps us define the probability for each mode.



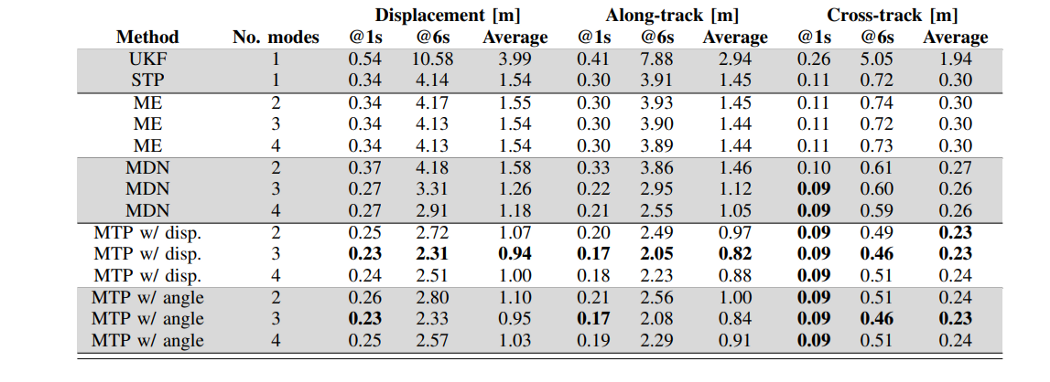
In the end, we sum up *class loss*and the *single-mode loss* using alpha as a hyper-parameter. Going out from formulas, what we are doing here is forcing the probability of the best matching mode **m∗** to be as close as possible to 1, and push probabilities of the other modes to 0. In other words, during training the position outputs are updated only for the winning mode, while the probability outputs are updated for all the modes. This causes each mode to specialize for a distinct class of actor behavior (going straight or turning) and successfully addresses the mode collapse problem that was shown before.

This algorithm also can be used with lane-following predictions. To make this happen, the authors added the 4th dimension to the raster image and keep all other the same. In this picture you can see lanes marked with pink color:

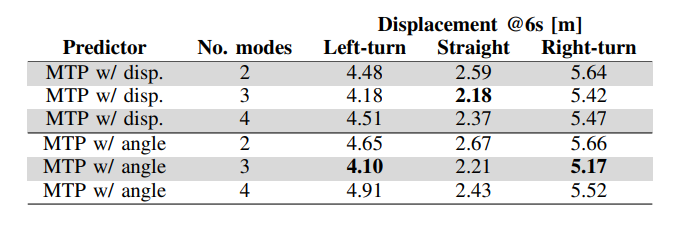


And now, let’s use the network and the new loss function and get the results. For this, the authors drove for 240 hours and collected the data from sensors, lidar, and radar. Then, they used this data to generate an actor-specific BEV raster image and used it to train their model. They used Tensorflow to build it and 16 Nvidia Titan X for 24 hours training with the batch-size of 64 images.

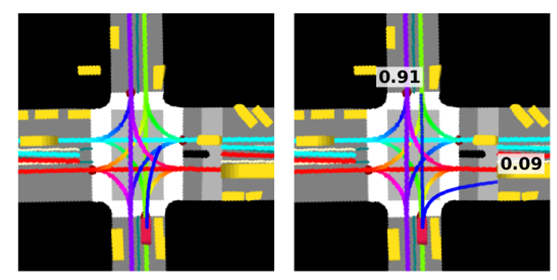
They used different metrics, such as displacement error (two coordinates) and along-track / cross-track (one coordinate) errors with different scope — 1-second scope, 6-second scope, and average between scopes. As a baseline authors used Unscented Kalman filter that forward propagates estimated state in time, Single-trajectory predictor from their previous paper, Mixture Density Network (Gaussian mixture over trajectory space), and multimodal model with two losses — *Mixture-of-Experts* and *Multi-Trajectory Prediction*.

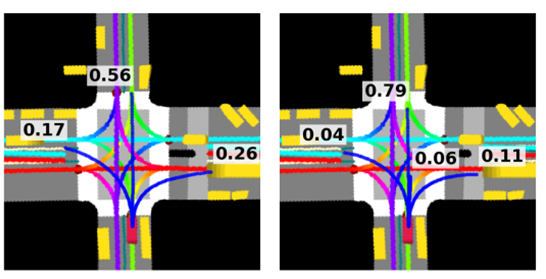


It is very interesting, that in short-term (1 second) all the models showed quite similar results. But when it went to the 6-second view, the best one was definitely a multimodal approach with *Multi-Trajectory Prediction Loss.*



Moreover, the authors decided to check a number of modes that show the best result, and it appears to be 3.





In the end, authors make the conclusion that their approach beats others using their own data representation, CNN model and unique loss. But what I think about this paper?

First of all, I am not sure, why the authors do not compare their solution with successful new ones, based on Bayesian Networks and Gaussian Processes. Moreover, they compare their model only with the model from 2000, model from 1994 and their previous model, that wasn’t compared to any else.

Second, why authors do not consider time changes? They did not use any information from the past. It was said, that they do not use RNN networks because it is impossible to make them multimodel, but why then not compare this unimodal approach with their CNN multimodal?

Third, why authors build their model on the top of MobileNet-v2? In their previous paper, another network showed better results. If they make experiments on which one is better in multimodal cases — why not include it to the paper?

In my personal opinion, there are also ways to improve their results — as I wrote before they can consider time changes, use each actor prediction to build other predictions and make the choice of mode numbers automatic, not manual. Still, with all these questions, their approach to improving their specific method with their specific data is quite interesting. In my opinion, the most useful thing in this paper is the multimodal approach itself — not only in autonomous driving but in any situation where we have a danger of “too generalized” solution.