CNN, CNN Encoder-RNN Decoder, and Pretrained Vision Transformers for Surrounding Vehicle Lane Change Classification at Future Time Steps

Neha Konakalla, Avrum Noor, Josh Singh
Department of Computer Science
Stanford University
{nkona, avrum, jsingh5}@stanford.edu

Abstract

Lane change (LC) classification at future time steps is key to ensuring the safety of passengers in vehicles, specifically autonomous vehicles. In this work, we propose a CNN Baseline, CNN-encoder RNN-decoder, and a fine-tuned pre-trained ViT model and demonstrate their performance on LC classification at future time steps. We trained and tested these models on the PREVENTION dataset [9]. We find that the baseline CNN model obtained a training accuracy of 86.2% and a testing accuracy of 80.87%. The baseline CNN was able to focus on lane markings as well as cars in the same, right, and left lane in front, a key aspect of predicting lane changes. The CNN encoder-RNN decoder model obtained 82.13% accuracy on the training data and 12.4% on the testing data. It seems as though the CNN encoder-RNN decoder predicts lane changes for all test examples. The ViT model with 10 and 4 frames obtained 84.51% and 82.13% accuracy on the training data and 81.23% and 81.1% accuracy on the the testing data, respectively. The ViT model seemed to only predict no LC for all test examples. The ViT models obtained the highest testing accuracy, but the CNN baseline actually predicted both LC and no LC which none of the other models did. Future work includes training and testing on a dataset where LC examples are more represented and/or changing the weighing of the losses depending on the classes.

1. Introduction

The enhanced safety benefits that autonomous vehicles provide over human drivers have led to the popularity of autonomous vehicles research and application. Lane change (LC) is a critical manoeuvre for the safety of vehicles in highways. Today, merge collisions accidents account for over 9% of all automobile crashes [5]. While the current state of autonomous vehicles have shown much performance gains at performing actions like lane changing, stopping, and accelerating, much work is still needed when it comes to reliably predicting lane changing of surrounding vehicles [13]. A reliable LC prediction model should be able to provide information regarding the lane change of surrounding vehicles at future time steps as early as possible in order to allow an autonomous vehicle's Advanced Driving Assistance Systems (ADAS) to pro-actively make driving decisions and avoid collisions.

While LSTM models can be used to perform lane change prediction tasks, they are limited by their shortcomings in learning spatial interdependencies [18]. An important aspect of predicting LC is the ability to model spatio-temporal relationships as surrounding vehicles that perform LC demonstrate complex spatio-temporal characteristics. To address these shortcomings, we set out to experiment with other architectures to understand how different architectures perform on lane change predictions and how they could potentially make up for the disadvantages of the LSTM.

We start with our baseline of a CNN-architecture [17]. We then train and test on a CNN encoder-RNN decoder architecture and a vision transformer (ViT) forecasting model. These models take driving data from a video camera mounted to the front of a car as input and output the predictions of whether the car in front, if present, is lane changing to the right, left, or staying in the same lane. These models are trained on around 30 mins ($\sim$ 20,000 frames) of video data and tested on 7 mins ($\sim$ 4,000 frames) of video data.

The baseline was able to perform 86.22% training accuracy and 80.87% testing accuracy and seemed to focus most of the weights on finding the lanes and the car in front. The CNN encoder-RNN decoder was able to perform 82.13% training accuracy and 12.4% testing accuracy and seemed to only predict LC. The ViT model was able to perform 84.51% training accuracy and 81.23% testing with an observation window of 10 frames and was able to perform 82.13% training accuracy and 81.1% testing with an observation window of 4 frames. The ViT model seemed to only predict no LC.

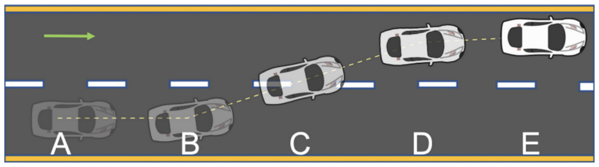

Figure 1. An illustration of lane change action. Illustration taken from [13].

2. Related Works

Most recently, Wirthmüller et al. (2021) [18] developed a system using LSTM-based RNNs to predict the time until LC occurs with a median of error of less than 0.25 seconds and as early as 3.5 seconds. However, the system has shown to be unreliable for predicting LC actions that occur at more than 4.5 seconds ahead in the future. Mozaffari et al. [14] attempts to address the shortcomings of using LSTM-based models to predict the likelihood of LC manoeuvres and the time-to-lane-change (TTLC) by proposing a Multi-Task Attention-based CNNs. The CNN allows spatial attention, which improves the feature extraction process by focusing on the most relevant areas of the surrounding lanes. However, the CNN's performance in short-term LC prediction is comparable to that of LSTM-based models and training CNN has proven to be more difficult. In this paper, we use a novel model that is a combination of CNN and RNNs/LSTMs (encoder-decoder networks) as a comparison for the ViT model and check to see if the ViT model has the potential to outperform a novel model that is designed to address the limitations of both CNNs and LSTMs.

Biparva et al. (2021) implemented four different two-stream pathways based approaches for this same task on the PREVENTION dataset, primarily using convolutional based networks with spatial and temporal channels. The four different approaches implemented and evaluated in [2] are Disjoint Two-Stream ConvNet [16], Two-Stream Inflated 3D ConvNet [4], Spatiotemporal Multiplier Networks [6], and SlowFast ConvNet [11]. For Two-Stream ConvNet, Two-Stream Inflated 3D ConvNet, and Spatiotemporal Multiplier Networks, the two stream pathways are a spatial stream in the form of individual region appearance and a motion stream in the form of dense optical flow across frames. On the other hand, for SlowFast ConvNet, the two pathways are obtained from region appearance, but taken with two different sampling rates (slow and fast). In [2] and this paper, LC prediction tasks are posed as video-recognition problems. By doing so, we are able to divide the task into a classification and regression sub-problem. They showed that Disjoint Two-Stream ConvNet and Spatiotemporal Multiplier Networks yield the best results for long observation horizons (time between first frame recorded and prediction time) while SlowFast ConvNet yields the best recognition performance from visual cues taken from short observation horizons. In [2], spatial and temporal channels are used primarily with convolutional based networks and in this paper, we hope to exploit these channels using ViT.

Additionally, there are existing transformer implementations, such as Gao et al. (2021) [7], that predicts lane change but only at the current time step and for the vehicle that the passenger is in. In [7], a span-based Transformer is used to extract features from front-view videos through span-based dynamic convolution, which captures both global and local dependencies. These features are then fed into an LSTM that is designed to both model long-term dependencies and features that occur simultaneously in order to detect the vehicle's lane changing behavior.

Other approaches for classifying and predicting LC are generative-based, such as Naive Bayes Classifiers [15], Bayesian Networks [10], and Hidden Markov Models [12]. RNN-based approaches are also becoming increasingly more common. In general, there are three different levels of modeling approaches for vehicle motion [3]: physical-based, where predictions only depend on vehicle kinematics, maneuver-based, where the motion of a vehicle depends on the maneuver of the driver, and intention-aware, where predictions take into account the inter-dependencies between vehicles. To make accurate LC prediction, models need to be intention-aware because the prediction of the trajectories of surrounding vehicles can be estimated more accurately if the inter-dependencies of lane changing vehicles are learned.

3. Dataset

We use the PREVENTION Dataset [9]. The PREVENTION Dataset is comprised of 5 separate recordings that make up 6 hours and 17 minutes of driving mostly on the freeway. This data was collected using 6 High-Speed FHD+ Bayer Cameras mounted to the front of the car, Lidar detectors stationed on top of the car, and radar detectors all around the vehicle. The PREVENTION Dataset provides users with multiple files that include lane changing data, lane labeling data, trajectories, and detections.

In our work, we specifically use the first recording taken (out of the five recordings) and split it for our train and test data. The first video is comprised of 37 mins and 47 kilometers of driving. These 37 mins of video were broken up into 23845 frames. These frames were originally RGB images of the resolution (600, 1920, 3). For training, each model created its training and testing data by sampling 5/6 of the frames for training and 1/6 for testing. This means that the training was made up of 19,871 frames and the testing was made up of 3,974 frames.

Our data preprocessing pipeline took raw camera footage from the form of a MP4 and turned it into numpy arrays of rgb values. The first step included splicing the raw video data into frames using openCV. The second step was running the openCV function imread on every frame which converts each frame to RGB matrix. In the CNN baseline and the CNN Encoder-RNN Decoder the RGB matrices was scaled down by 25% and for the ViT pre-trained Transfomer, the RGB matrix was scaled down by 50%. These scalings were used because the CNN baseline and the CNN-RNN model were running into memory issues while the ViT model was able to handle the more amounts of data.

The specific inputs to CNN baseline and CNN-RNN model was a horizontal concatenation of the frames' scaled RGB matrices from an observation window. Our observation window was the current and the previous three time frames. Due to memory constraints, not present in the ViT model, we kept the observation window to four frames.

For the ViT model, we ran it twice with two different observation frames. One observation window was the current and the previous nine time frames and the other observation was the current and the previous three time frames, like the other models. We passed in the horizontally concatenated RGB matrix of the 4 or 10 frames into a specific pytorch feature extractor created by HuggingFace called ViTFeatureExtractor from the Transformers Library. This feature extractor returns a augmented version of the concatenated RGB matrix to pass into the pre-trained ViT Transformer.

Additionally, the lane classifications (e.g. left lane change, right lane change, no lane change) were assigned to an interval of frames in the original dataset, so preprocessing had to performed to convert this format into a mapping of each individual frame number to its corresponding ground truth lane change change class.

4. Methods

There are three distinct models (one run twice with different observation windows) that we have trained to predict the lane change class at 10 frames (1 second) ahead of the current frame. Each takes in a single image tensor that is created by concatenating all of the images in the observation window horizontally, by extracting the 4 or 10 frames up to the current time step, converting each frame into an RGB matrix, performing scaling on the matrix based on model architecture, and then concatenating the 4 or 10 matrices in the horizontal dimension. Then, depending on the model, an additional featurization of the concatenation is made before being passed in as input to the model.

Next, in order to load batches of data into each model at a time, we made use of Pytorch Dataloaders on the training set and test set, which have a split ratio of 5:1.

For the criterion, all three of the models use Cross Entropy Loss because we are measuring the accuracy of multiclass classification. Specifically, Cross Entropy Loss has the following equation:

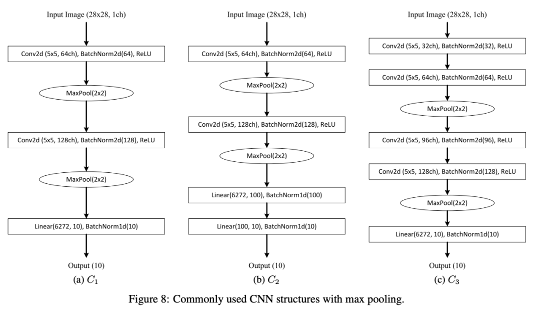
Input Image (28x28, 1ch)
Conv2d (5x5, 64ch), BatchNorm2d(64), ReLU
MaxPool(2x2)
Conv2d (5x5, 128ch), BatchNorm2d(128), ReLU
MaxPool(2x2)
Linear(6272, 10), BatchNorm1d(10)
Output (10)
(a) C_1
Input Image (28x28, 1ch)
Conv2d (5x5, 64ch), BatchNorm2d(64), ReLU
MaxPool(2x2)
Conv2d (5x5, 128ch), BatchNorm2d(128), ReLU
MaxPool(2x2)
Linear(6272, 100), BatchNorm1d(100)
Linear(100, 10), BatchNorm1d(10)
Output (10)
(b) C_2
Input Image (28x28, 1ch)
Conv2d (5x5, 32ch), BatchNorm2d(32), ReLU
Conv2d (5x5, 64ch), BatchNorm2d(64), ReLU
MaxPool(2x2)
Conv2d (5x5, 96ch), BatchNorm2d(96), ReLU
Conv2d (5x5, 128ch), BatchNorm2d(128), ReLU
MaxPool(2x2)
Linear(6272, 10), BatchNorm1d(10)
Output (10)
(c) C_3
Figure 8: Commonly used CNN structures with max pooling.

Figure 2. Common Image Classification Architectures [1]

$$L_i = -log(\frac{e^{(f_y)_i}}{\Sigma_j e^{f_j}})$$

Inside the log expression, the normalized probability assigned to the correct class label y_i for a given input, so Cross Entropy Loss aims to minimize the negative log likelihood of of the correct class (in our case, left/right/no lane change). This is the same as performing maximum likelihood estimation on the correct class, resulting in the model learning to maximize predicting the correct y_i class in our multi class problem.

Each of the models resulting outputs are representing raw, unnormalized scores for each of the three classes, the first entry of the tensor being no lane change, the second being left lane change, and the third being right lane change. These logits go through a softmax function to convert each logit into a probablity that the model predicts for each class for the given example. The softmax function is defined below as follows:

$$f_j(z) = -log(\frac{e^{z_j}}{\Sigma_k e^{z_k}})$$

4.1. CNN Baseline

First, we developed a baseline using a CNN architecture inspired from the An et al. (2020) paper classification architecture [1] because it achieved state of the art results on the MNIST Dataset (see figure 2 for more details). This CNN approach is similar to most prior works, which we hope to contrast with our other novel approaches. We chose to implement a more basic CNN architecture as we are dealing with a limited data size, and as an initial step to improve results upon. The architecture is as follows: First, in order to extract relevant features for prediction from the images, we pass the input image into a 2D convolutional layer, with 3 input channels, 32 output channels, kernel size of 3, and stride of 1. Then, in order to provide a nonlinear activation to provide more useful transformations of the features, we take the ReLU of this output, which is thereafter passed into

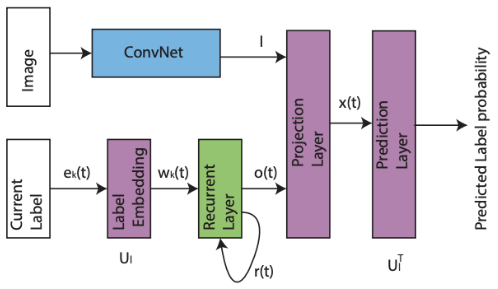

Figure 3. Overall Architecture for the CNN Encoder-RNN Decoder Model from [17]

another 2D convolutional layer for further feature extraction, with 32 input channels, 64 output channels, kernel size of 3, and stride of 1. Then, the ReLU of this output is taken, which is then passed into the MaxPool2D function with a kernel size of 40. We added max pooling here to reduce size of the neural network, which grew too large even with convolutional layers. Then, we apply dropout to prevent overfitting through probabilistic removal of inputs to this layer. Then, we flatten the result, to be able to feed it with the right shape into a fully connected layer with 9024 input features and 128 output features. Then, we apply dropout again and pass the output through another fully connected layer with 128 input features and 3 output features. These three resulting output features are logits, which are then converted into a probability.

4.2. CNN Encoder-RNN Decoder

In this model, we used a CNN encoder and RNN decoder for multilabel classification (see figure 3). The CNN encoder aims to learn images' semantic representation while the RNN decoder deals with the image/label relationship and label dependency. The convolution layers within the CNN captures feature of images while the recurrent layer within the RNN captures information from previous predictions in order to model the label co-occurrence dependency in a joint image/label embedding space. We heavily relied on Wang et al (2016) [17] and code from Lin Zhipeng (https://github.com/Lin-Zhipeng/CNN-RNN-A-Unified-Framework-for-Multi-label-Image-Classification). Despite the fact that the RNN framework is not explicitly designed for attention-based tasks, the CNN Encoder-RNN Decoder model is able to attend to different segments of the image regions when predicting different labels, which can prove useful for modeling moving objects such as vehicles. [17].

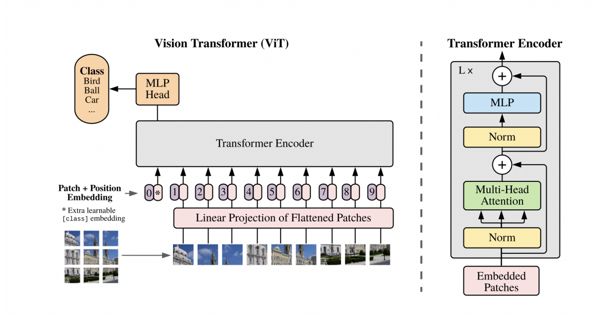

Figure 4. Overall Architecture for VIT

4.2.1 CNN Encoder

For the CNN Encoder, a ResNet152 model pretrained on ImageNet dataset from torchvision.models is used. The last fully connected layer is deleted and replaced with a linear layer that has an output of size embed_size, which is set to three given that there are three classes in the lane changing (Right Shift, Left Shift, No Shift/Same Lane).

4.2.2 RNN Decoder

The RNN Decoder takes in a set of embeddings and feeds them into an LSTM layer, which maintains a hidden state to learn co-occurrence information within the labels. The output of the LSTM layer then is connected to a linear layer, which gives out the probability of a label at a given time step.

4.3. ViT Pretrained Transformer

Our next approach was to finetune Vision Transformer (ViT), a transformer implementation that does not rely on CNN image processing to perform image classification. ViT was pretrained on ImageNet, where greater than 14 million images are annotated with bounding boxes around objects, indicating objects present in the image, to be used for classification tasks. Images that are trained and finetuned on by the model must first pass through the ViT feature extractor, which splits each image into non-overlapping patches. We used a patch size of 16 x 16, the same size as used in the original ViT model training. Then, these patches are linearly embedded and a CLS token is added to mark the creation of an entire image representation vector. Then, after adding an absolute position embedding, the result is a single example to be fed into the model. The model architecture itself consists of a transformer encoder, where image embedded patches are first normalized. Then, a multiheaded attention layer is used, which helps better attend to the input sequence in different ways, helping capture a greater variety of patterns (e.g. both long-term and short-term dependencies). The next step is to merge and normalize the multiheaded attention output and then pass the result through a multi-layer perception where the final layer of the perceptron takes the softmax of the logits of the classes. In order to set up the model for finetuning, we first downloaded the pretrained ViT model with a patch size of 16x16 and image resolution of 224x224 from the Huggingface library. Then, we modified the number of output classes in the final layer to be 3 for finetuning, as the model originally trained on 21K classes. We finetuned ViT twice, once using 4 historical frames to predict 10 frames ahead, and another time using 10 historical frames to predict 10 frames ahead.

5. Experiments, Results, and Discussion

5.1. Hyperparameters

For the CNN Baseline and ViT models, we trained this model for 11 epochs with a batch size of 32, and Cross Entropy Loss. We trained for 11 epochs because after looking at the training accuracy and loss, 11 epochs was more than enough time for those to converge. We used a batch size 32 because Kandel et al. (2020) [8] states that it is a good default value that will quicken the computation of the network but decrease the number of updates required to reach convergence. Cross Entropy Loss is used as the loss function because it is the most often used loss function for multi-class classification tasks. We used a SGD optimizer with a learning rate of 0.01, a momentum of 0.9, and kept the rest of the pytorch SGD defaults. We used this optimizer and values in accordance to the An el al paper [1] and for the ViT to stay consistent with the baseline.

For the CNN Encoder-RNN Decoder, we trained this model for 5 epochs with a batch size of 32, and Cross Entropy Loss. We choose 5 epochs because accuracy and loss seemed to converge. We used a Adam Optimizer a learning rate of 0.01 and keep the rest of the pytorch Adam defaults. We used Adam Optimizer with accordance to Wang et al [17].

5.2. Results and Discussion

For test and evaluate our models, we calculate the accuracy and true/false positive/negative ratios for quantitative understanding and build saliency mappings and run our models with different observation windows for qualitatively understanding our models.

To calculate accuracy, the logits from the model and ground truth labels are needed. The logits from the model are then softmaxed and then the class with the max probability is found. Then it is checked to see if the class with the max probability is the ground truth. This is done for all the outputs and labels and an overall accuracy is achieved. Accuracies can be found in Table 1.

True/False positives/negatives ratios are important to analyze to understand how the model is doing in specific classes. Accuracy tells an overall story, but this analysis is needed specifically because we need to understand in predicting which class does the model do the worst. Also, there are way more examples of no lane change than lane change, because cars spend more time in lanes than they spend changing lanes, so these metrics can offer more insight that bypasses this shortcoming. We set out to understand more about lane change behavior, so we must understand accuracy of predicting lane changes vs no lane changes.

In this analysis, we abstract away the difference between right and left lane changes and just have to possible labels: lane change or no lane change. This is done as there are very little amount of examples of left and right lane changes so to better analyze these results, we need to just think of lane changes as a whole.

We define our true positive ratio as, out of all the correct predictions, the percent that were examples with a LC (TP/(TP+TN)). The true negative ratio is, out of all the correct predictions, the percent that were examples with no LC (TN/(TP+TN)). The false positive ratio is, out of all the examples that were incorrect, what percent incorrectly predicted that a LC occurred (FP/(FP+FN)). The false negative ratio is, out of all of the examples that were incorrect, what percent incorrectly predict that no LC occurred (FN/(FP+FN)). These ratios can be found in Table 2.

5.2.1 CNN Baseline

The CNN baseline was able to perform 86.2% correctly on the training data and 80.9% on testing data. These are relatively high frequencies that shows that the model predictions are right for majority of the labels. This could mean that the model is doing a great job discriminating between LC or no LC or it could be due to a relatively uniform correct labels of one class.

True/False Positive/Negative ratios help with understanding more about how the predictions of baseline do with regards to specific classes. Table 2 indicates that the CNN baseline, when it got the prediction wrong, was much more likely to falsely predict no lane change (97.99%) than falsely predict lane change (2.01%). These stats coupled with the fact that the dataset is mostly no LC means that the model just be heavily biasing to learning to predict no LC most of the time. This problem can be remedied with weighing the losses of false negatives higher than false positives and/or creating a different dataset that has lots of lane changes in it.

Saliency maps to show feature attribution and are used to show the strength for each pixel contribution to the final output. They are found by feeding the model into a TruLens module. This module uses backpropagation to compute the gradient of the losses regarding the pixels of the images to learn more about the pixels' importance.

Model	CNN Baseline	CNN RNN EncDec	ViT 10 Frames	ViT 4 Frames
Training Accuracy	86.22%	82.13%	84.51%	82.13%
Testing Accuracy	80.87%	12.4%	81.23%	81.1%

Table 1. Models' Training and Testing Accuracies

Model	CNN Baseline	CNN RNN EncDec	ViT 10 Frames	ViT 4 Frames
True Positive Ratio	1.71%	100.0%	0.0%	0.0%
True Negative Ratio	98.29%	0.0%	100.0%	100.0 %
False Positive Ratio	2.01%	100.0%	0.0%	0.0%
False Negative Ratio	97.99%	0.0%	100.0%	100.0%

Table 2. Models' True Positive and Negative Ratios and False Positive and Negative Ratios

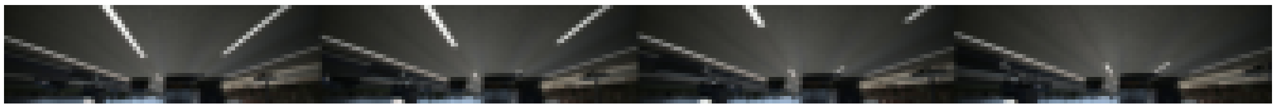

Figure 5. Orginal Observation Window

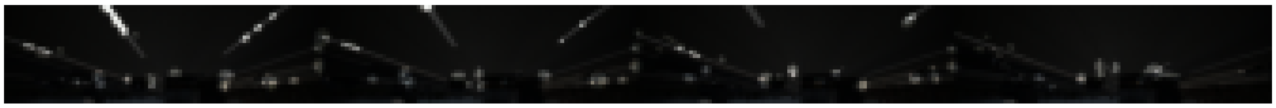

Figure 6. Mapping of most important pixels to the Model

As is seen in the saliency map for the CNN baseline (see Figs 5 and 6), there are some key aspects of the image the model takes a look at. Please note the images are upside down as that is how the dataset was fed in. First, the model pays a lot of attention to the lanes of the freeway which is important because to learn and predict lane change behavior, the model must know where the lanes are. Second, the model pays attention to three clusters of space. First, it pays attention to the space where the car would be if it was in front of it the camera. The other two spaces are where the car would shift to if the car was lane changing. In this way the model is able to learn the lanes, if there is a car in front, and if the place where the cars would lane change to are empty. These are extremely important details as there can only be a lane change if there is a car in front and if the other lanes are empty.

5.2.2 CNN Encoder-RNN Decoder

The CNN Encoder-RNN decoder was able to perform 82.13% correctly on the training data and 12.4% testing data, respectively. These accuracies are not as high as the CNN baseline, especially the testing accuracy.

Table 2 indicates that the model only predicted LC. This can be seen as the True Positive and False Positive Ratios are 100% which means that out of all the incorrect and correct examples, LC was always predicted. This model is deeply flawed. We believe that this could be the case because this model is not able to take in a lot of information from each frame as it focuses on one part of the image. In other words, the CNN-RNN can only focus on one part of the image at a time, so it is taking in less information than it could.

5.2.3 ViT 4 and 10 Frames

The ViT 10 frames and the ViT 4 frames were able to perform 82.13% and 84.51% correctly on the training data and 81.1% and 81.23% on testing data, respectively. The testing accuracy for these models are the highest while the training accuracies are not as high as the CNN baseline.

Table 2 indicates that the ViT-4 and ViT-10 models only predicted no LC. This can be seen as the True Negative and False Negative Ratios are 100% which means that in all of the incorrect and correct examples, no LC was predicted. This shows a clear flaw in this model, one that shows that this model is not useful, even though it has such a high accuracy. The dataset should be made more even between LC and no LC and the weighting of the losses of the different classes should be changed so that it actually starts predicting LC.

As ViT is a pretrained model, we weren't able to use TruLens to build a Saliency Map. Instead, we ran the model with two different observation windows to see how that affected the accuracies. It seems as though the testing accuracy was about the same while using 4 frames as it was for using 10 frames, hinting that the other 6 frames do not provide much information about lane changing. This means that lane changing is more instant or the model is not learning non-instant important indicators of lane changing that happen for a while before lane changes occur - like blinker lights.

5.2.4 Overall Analysis

Overall, the ViT models performed the best on the test data and the baseline outperformed all models in training accuracy, true/false positive/negative ratios, and focused on the exact parts of the image that it should have. It is important to note that even though the ViT outperforms the baseline test accuracy wise, it is likely that the CNN baseline is a better model to employ because it actually is able to predict both LC and no LC. The high ViT test accuracy is just a byproduct of there being lots of examples of no LC.

The testing accuracy of the CNN baseline and the ViT models are on par with the results found by Biparva et al, who used CNN based architecture [2]. They found 83.61% testing accuracy using 20 frames as an observation window on their baseline model to classify 10 frames ahead. Our results are on par, even though the dataset we used is much smaller than theirs.

In order to better all the models, more experiments should be run with a dataset that is made up of more LC - then we can understand more about how this model improves in classifying LC. The other two models do not seem to be learning how to discriminate between LC and no LC and therefore would not be as good of choices compared to the baseline.

6. Conclusion

The ViT pretrained model obtained the best testing accuracy, while the CNN Baseline is most likely the best model for prediction LC vs no LC. The ViT model always predict no LC due to the high prevalence of no LC examples. The CNN baseline was able to extract important features from the input data, such as lane markings, and cars ahead in same, right, and left lane, to make more educated predictions of whether the car infront will lane change. Even though the CNN was the only model that was able to predict both no LC and LC, it still performed poorly on correctly predicting LC. The CNN encoder-RNN decoder, despite having a training accuracy on par with the other two models, had a terrible testing accuracy. It only predicted LCs to occur.

In future work, it would be interesting to weigh the loss of the different labels higher. For example, incorrectly classifying an actual lane change as no lane change weighted higher than classifying no lane change as a lane change. This way we can make up for the fact that the dataset is so skewed: the majority of the video is of no lane change. We should also experiment with predicting 5 or 20 frames ahead instead of just 10. We will be able to analyze how early cars should be able to detect lane changes. We may also try a different pre-trained transformer as this transformer is not pre-trained on car related objects/actions. Also, we can experiment with object detection in our preprocessing to mask everything in the frames except for the blinkers. Then the blinkers can be passed in to the CNN and then CNN will hopefully learn that when the color changes that indicates lane change.

Overall, the ViT model outperform the CNN baseline, but the CNN baseline seems to be the better model as it is able to (albeit poorly) discriminate and predict both LC and no LC. Further exploration with different datasets, observation windows, transformers, and object detection may help improve accuracy of Lane Change classification at future time steps.

References

[1] Sanghyeon An, Minjun Lee, Sanglee Park, Heerin Yang, and Jungmin So. An ensemble of simple convolutional neural network models for mnist digit recognition, 2020. 3, 5

[2] Mahdi Biparva, David Fernández-Llorca, Rubén Izquierdo-Gonzalo, and John K. Tsotsos. Video action recognition for lane-change classification and prediction of surrounding vehicles, 2021. 2, 7

[3] Julian Bock, Robert Krajewski, Tobias Moers, Steffen Runde, Lennart Vater, and Lutz Eckstein. The ind dataset: A drone dataset of naturalistic road user trajectories at german intersections, 2019. 2

[4] João Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4724–4733, 2017. 2

[5] Greg Coleman. Lane change accidents: Changing lanes dangerously, Jan 2022. 1

[6] Christoph Feichtenhofer, Axel Pinz, and Richard P. Wildes. Spatiotemporal multiplier networks for video action recognition. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7445–7454, 2017. 2

[7] Jun Gao, Jiangang Yi, and Yi Lu Murphey. A lane-changing detection model using span-based transformer. In *2021 33rd*

Chinese Control and Decision Conference (CCDC), pages 2733–2738, 2021. 2

[8] Mauro Castelli Ibrahem Kandel. The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset, 2020. 5

[9] R. Izquierdo, A. Quintanar, I. Parra, D. Fernández-Llorca, and M. A. Sotelo. The prevention dataset: a novel benchmark for prediction of vehicles intentions. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 3114–3121, Oct 2019. 1, 2

[10] Dietmar Kasper, Galia Weidl, Thao Dang, Gabi Breuel, Andreas Tamke, Andreas Wedel, and Wolfgang Rosenstiel. Object-oriented bayesian networks for detection of lane change maneuvers. *IEEE Intelligent Transportation Systems Magazine*, 4(3):19–31, 2012. 2

[11] Myeongjun Kim, Taehun Kim, and Daijin Kim. Spatio-temporal slowfast self-attention network for action recognition. In *2020 IEEE International Conference on Image Processing (ICIP)*, pages 2206–2210, 2020. 2

[12] Peng Liu, Arda Kurt, and Ümit Özgüner. Trajectory prediction of a lane changing vehicle based on driver behavior estimation and classification. In *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 942–947, 2014. 2

[13] Vishal Mahajan, Christos Katrakazas, and Constantinos Antoniou. Prediction of lane-changing maneuvers with automatic labeling and deep learning. *Transportation Research Record*, 2674(7):336–347, 2020. 1, 2

[14] Sajjad Mozaffari, Eduardo Arnold, Mehrdad Dianati, and Saber Fallah. Early lane change prediction for automated driving systems using multi-task attention-based convolutional neural networks. *CoRR*, abs/2109.10742, 2021. 2

[15] Julian Schlechtriemen, Andreas Wedel, Joerg Hillenbrand, Gabi Breuel, and Klaus-Dieter Kuhnert. A lane change detection approach using feature ranking with maximized predictive power. In *2014 IEEE Intelligent Vehicles Symposium Proceedings*, pages 108–114, 2014. 2

[16] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. *CoRR*, abs/1406.2199, 2014. 2

[17] Jiang Wang, Yi Yang, Junhua Mao, Zhiheng Huang, Chang Huang, and Wei Xu. Cnn-rnn: A unified framework for multi-label image classification, 2016. 1, 4, 5

[18] Florian Wirthmüller, Marvin Klimke, Julian Schlechtriemen, Jochen Hipp, and Manfred Reichert. Predicting the time until a vehicle changes the lane using lstm-based recurrent neural networks. *IEEE Robotics and Automation Letters*, 6(2):2357–2364, 2021. 1, 2