CS455 Final Project

Julia Abdel-Monem

March 31, 2025

Abstract

1 Introduction

Background

Tailgating

According to the NHTSA in 2022, speeding accounted for 29% of accident related fatalities, and 7.8% of all fatalities involved at least one distracted driver [for Statistics and Analysis, 2024]. While there are strategies to avoid crashes with an inattentive or aggressive driver, three studies released by the highway loss data institute (Subaru EyeSight, Kia Drive Wise, and Honda FCW/LDW) demonstrate that crash avoidance features may reduce the number of collision claims made, with young drivers experiencing the most benefit [HLD, 2019, HLD, 2021b, HLD, 2021a]. While the elderly population must still be considered, this trend in age may be explained by overall trends [for Statistics and Analysis, 2024]. Newer cars are more expensive, so the goal of this project is to develop an app to help lower the financial barrier of entry from a modern car to a relatively recently made smartphone.

Semantic Segmentation

Semantic segmentation was chosen since it provides an accurate width, without being overly complex to use. An object detection model such as YOLO would be useful in finding which cars are in front of the camera, but an accurate width requires an accurate edge to read against. instance detection may be useful for this, but it may have more overhead compared to semantic segmentation, where individual cars can be isolated by the width of the lane.

Neural Network: Design and Training

The SegFormer

The SegFormer is a neural network designed to use vision transformers in semantic segmentation, using multiple transformers using an efficient self attention algorithm and a simple MLP decoder [Xie et al., 2021]. It builds upon the original patch transformer encoder described in [Dosovitskiy et al., 2020]. This hierarchical transformer encoder extracts both coarse and fine features. The original SegFormer paper provides five different levels of preconfigured hyperparameters, each one taking more computational power but resulting in a higher IoU.

Intersection over Union

To test the model's accuracy, the metric `Intersection over Union`, also known as IoU was chosen. Since there are only two classes that the model is being trained for (Car or Not Car), the BinaryIoU keras metric was chosen, comparing against both the 'Car' class (IoU_1) and the mean of both (mIoU). It is worth noting that the original design and experiment used mean IoU to validate training.

$$\text{IoU}_i = \frac{\text{True Positives}_i}{\text{True Positives}_i + \text{False Positives}_i + \text{False Negatives}_i} = \frac{T_i \cap P_i}{T_i \cup P_i}$$

$$\text{mIoU} = \frac{1}{c}\sum_{i=0}^{c}(\text{IoU}_i)$$

2 Methods

Neural Network

Implementation Details

The neural network was implemented from scratch using Tensorflow and Keras in Python, based off of the SegFormer paper [Xie et al., 2021], the Vision Transformer paper [Dosovitskiy et al., 2020], and the original Attention paper [Vaswani et al., 2017][1]. This was done in order to use Tensorflow and Keras, as there are no working implementations of a SegFormer in Keras 3[2]. The original paper implemented their neural network in *mmsegmentation* and PyTorch, and was one of the sources I used to implement my own, especially when it came to implementing the efficient self attention.
This project overwhelmingly used classes that inherited from `keras.Model` for every single layer since Keras does not permit multiple layers of inheriting `keras.layers.Layer` (as in, the `TransformerBlock` must be a `keras.Model` if it uses any layers implemented by the end user, or it will not "see" or train those weights).

Dataset

The dataset was trained on a subset Berkeley DeepDive 100K dataset, modified to only include one class. There is no existing implementation of this dataset for Tensorflow, so to allow the efficient use of the dataset, a Tensorflow dataset was constructed using `tfds`. This dataset was chosen for its size and spread among a diverse set of environments.

Training (and Finding the Correct Loss Function)

Finding the correct loss to train this model was the most complex part of this project by far, and the major trials will be shown in the results section later on.

[1]*Note to grader, this took the majority of the project time. I did not know what I was getting myself into.*

[2]I did find one written in Keras 2, which may have been upgraded to Keras 3, but I could not get it working. You can find it at github.com/IMvision12/SegFormer-tf/

Binary Crossentropy

Binary Crossentropy compares the difference between two probability distributions for a given random variable. This loss function was used in the original SegFormer paper.

$$L_{CE}(y,t) = -\sum_{n=1}^{N} \log(t_n \cdot y_n)$$

For a general semantic segmentation problem, this loss function has many advantages as a "Pixel Level" loss function [Azad et al., 2023]. However, as shown in the results, this did not work for this problem, due to the inbalance of importance between the two classes. Moving forward I chose to use a "Region-Level" loss function.

Dice Loss

Dice loss seemed very promising initially, as it the directly related Dice Coefficient measures the similarity between two sets of data, and is similar to the Intersection over Union. The dice coefficient is written as:

$$\text{Dice Coefficient} = \frac{2|Y \cap T|}{|Y| + |T|}$$

Similar to mIoU, when there is more than one class, the average of all the losses for each class is used as the loss. Since there are two classes, and one class is significantly more important to identify than the other, this loss function struggled to provide accurate feedback, as it weighed the IoU of the background class the same as the target class[3].

Tversky Loss

Enter Tversky Loss, the function that this project settled on. It allows the independent weighting of false positives and false negatives. It is based on the Tversky index, which is defined as[Azad et al., 2023]:

$$\text{Tversky Index} = \frac{|Y \cap T|}{|Y + T| + \alpha|Y/T| + \beta|Y/T|}$$

[3]This was identified by monitoring the IoU of each class individually, and noting that the average IoU kept rising and the loss kept falling, but the target class never rose higher than 0.01.

Where α is the weight for false positives, and β is the weight for false negatives. This loss function allows for these two extra hyperparameters to be optimized to maximize the correlation between loss and positive IoU.

Computational Power

At the offset of this project, I initially planned to train this neural network on my own graphics card (a GTX 1070). This caused this project to run into many issues and slowed down training significantly as I would consistently run out of memory on the GPU. So, I conceeded and chose to use Google Colab instead, paying $10 for 100 compute units so that I could use a Nvidia T4, with 16GB of VRAM and dedicated neural network compute units.

3 Results

Binary CrossEntropy, Binary IoU, B0

This combination somehow resulted in the AdamW optimizer landing on a mean IoU of 0.5, with one class at 0.9 and the other near 0.001. It somehow found the one location in the entire training dataset where no cars are.

Dice Loss, Binary IoU, no axis set, B0

Dice Loss, Binary IoU, Downscaled images, B2

Tversky Loss, Binary IoU, Downscaled images, B0

4 Conclusion

As i'm writing this, I'm hoping my model reaches a high enough accuracy to work for the final. But what killed this project ultimately is the lack of detail on the original paper on how they pretrained their model. They spend a total of one sentence on pretraining, which is unhelpful to say the least. I've run out of time to complete this project.

With that being said, I've learned a ton about transformer based neural networks and semantic segmentation. If I had time to pretrain the encoder, I'd design a pretext task based on the ImageNet-1K dataset. Pretraining the encoder may be outside of my current knowledge base, however. I wouldn't use tensorflow or keras, and would probably use the framework that best supports the model.
This was a fun challenge.

References

[HLD, 2019] (2019). Impact of Subaru collision avoidance features on insurance losses by rated driver age. *Highway Data Loss Institute*, Vol. 36, No 25.

[HLD, 2021a] (2021a). Impact of Honda Accord collision avoidance features on claim frequency by rated driver age. *Highway Data Loss Institute*, Vol. 38, No. 13.

[HLD, 2021b] (2021b). Impact of Kia Drive Wise collision avoidance features on insurance losses by rated driver age. *Highway Data Loss Institute*, Vol. 38, No. 14.

[Azad et al., 2023] Azad, R., Heidary, M., Yilmaz, K., Hüttemann, M., Karimijafarbigloo, S., Wu, Y., Schmeink, A., and Merhof, D. (2023). Loss functions in the era of semantic segmentation: A survey and outlook.

[Dosovitskiy et al., 2020] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *CoRR*, abs/2010.11929.

[for Statistics and Analysis, 2024] for Statistics, N. C. and Analysis (2024). Overview of motor vehicle traffic crashes in 2022. *Traffic Safety Facts Research Note. Report No. DOT HS 813 560.*

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. *CoRR*, abs/1706.03762.

[Xie et al., 2021] Xie, E., Wang, W., Yu, Z., Anandkumar, A., Álvarez, J. M., and Luo, P. (2021). Segformer: Simple and efficient design for semantic segmentation with transformers. *CoRR*, abs/2105.15203.