Dangerous driving situation detection through on-device computer vision

Aljosa Koren, Zan Gostic

ABSTRACT

Driver's assistance systems are becoming more popular each year. They greatly improve the overall safety in traffic. However, for some drivers they are unreachable as they cannot afford them. We managed to produce an application that uses computer vision and can work on a smartphone with a camera. Our application presents an affordable driver's assistance system that can help improve the pedestrians' safety.

1. INTRODUCTION

In this article we will present our work in making an application for the Mobile Sensing course addressing pedestrian detection with the phone's camera placed on the windshield of the vehicle. We made a driving assistant application, which alerts the driver when it detects pedestrians in dangerous situations on the road with a beeping sound. The phone needs to be placed on the windshield horizontally and the main camera should be focused on the road. When the application detects pedestrians crossing the road in front of the vehicle, it alerts the driver by producing a distinct sound alert, which urges a response of the driver to pay more attention to the road.

More than 7000 pedestrians are killed yearly, about one every 75 minutes. [1] With our application we are planning on reducing this enormous number of casualties. As many car owners cannot afford newer cars with additional expensive equipment, we are planing that with our application anyone with a smartphone could download our application and make roads safer.

2. RELATED WORK

In [2] context aided pedestrian detection is discussed. Authors have used laser scanner, computer vision and inertial sensors for the application. In [3] they discuss a prototype system used for pedestrian detection from on board of a moving vehicle. To achieve this they used a two step approach with the first using contour features in a hierarchical template matching approach to efficiently 'lock' on to the candidate features. The second step utilizes the richer set of intensity features in a pattern classification approach to verify the candidate solutions. In [4] an object detection system that can be automatically trained to detect objects of a certain class is discussed. They apply this system to detect pedestrians.

3. IMPLEMENTATION & EXPERIMENTS

3.1 Data

Our main data is the CityPersons dataset [5] which is a new set of annotations targeting pedestrian detection. It is based on the Cityscapes dataset [6, 7] which contains stereo images recorded in 27 European cities. Cityscapes dataset contains 5000 images with fine pixel-level annotations and 20000 images with coarse semantic labels. CityPersons contains only the fine pixel-level annotated images with 5 different classes: ignore regions, pedestrians, riders, sitting persons, other persons with unusual postures and a group of people. Ignore regions represent hard negatives or areas that can be easily misinterpreted as humans for example posters with people on it, reflections etc. The dataset contains 35000 persons (20000 unique) and 13000 ignore labels. Images contain an average of 7 persons. 83% of the annotated persons are pedestrians, riders represent 10% of the labels and sitting persons 5%. Images have a height of 1024 and width of 2048 pixels.

Additionally, we created a script that allowed us to add annotations to each marked pedestrian. We decided on 4 labels. First label annotates the ignore regions. Second is for pedestrians that are safely on the sidewalk or very far away. This annotation is called 'safe pedestrian'. The third label marks pedestrians that look like they plan on crossing the road, the cyclist driving near and pedestrians crossing the road down the road. We call this annotation 'be careful'. The last label is for the pedestrians that are crossing the road in front of the vehicle and the car would hit them if it did not brake immediately. We call this annotation 'brake'. We can than classify the images in one of four classes by giving them the class of the most dangerous individual on the image.

First we split the training set into train set and the validation set. We decided to use images from two

towns as the validation set and other 16 as the train set. In the test set there were images from 3 never seen before cities. Train set contained 2610, validation set 365 and test set 500 images. In Figures 1, 2 and 3 we can see that the train, test and validation dataset have similar distributions, as they all contain majority of images containing pedestrians that are safely on the sidewalks. However, validation dataset contains larger percentage of images where there are no pedestrians in sight and test set contains a larger percentage of images where pedestrians are in danger if the car doesn't brake. Images were manually annotated so there might be mistakes and the subjective bias integrated in the labels. In Figure 4 we can see four examples of newly labeled data, where pedestrians are labeled with four different classes.

Secondly, we split the data in a different way. The ratio of images stayed the same as before for train, validation and test set, however, we used images from all cities in all sets. This means that in this case the model might have already seen an image from the same street during training that it got in the test set (but not exactly the same image). We split the dataset using stratification of the labels, so classes had the same representations in all cases.

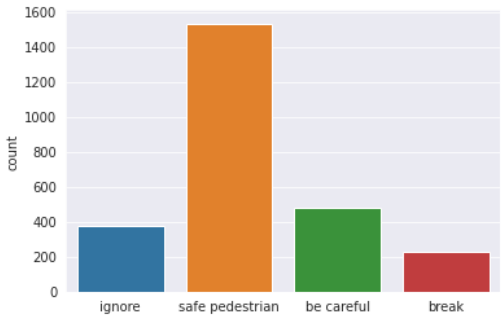

Figure 1: Train dataset distribution.

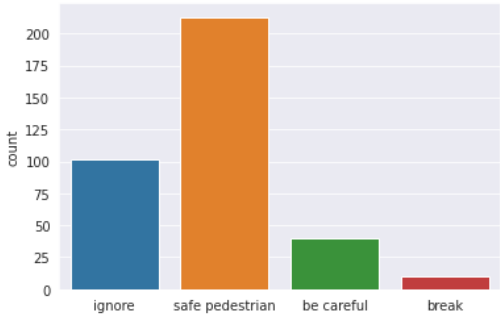

Figure 2: Validation dataset distribution.

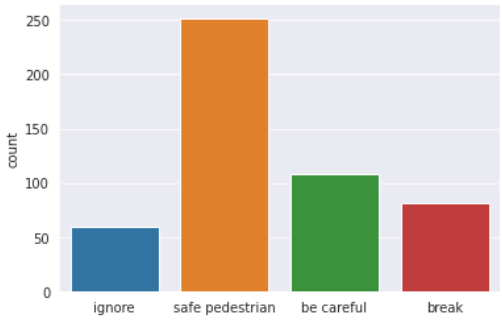

Figure 3: Test dataset distribution.

3.2 Computer vision

Firstly, we tried using the pretrained MobileNetV2 [8] on the ImageNet dataset [9]. The first model $PD_{v1}^{MobileNetV2}$ has frozen pretrained weights on the base MobileNet network. On top we added additional global average pooling layer and 3 dense layers with 64, 32 and 4 neurons. Between the dense layers we added dropout layers with 20% dropout rate for regularization. Two layers have ReLU activation function while the last has the softmax activation function. As per learning rate we used 0.00001, with Adam optimizer and sparse categorical cross-entropy. We decided to use the batch size of 8. Firstly, we reached the problem of unbalanced dataset. The model reached the accuracy of 50% which was close to the majority classifier. We continued by making the class weights of the minority classes bigger. By doing this we got a better accuracy by 1% and also the classifier produced greater variety of outputs instead of just the majority class. We also introduced data augmentation to increase the dataset without the need of classifying new images and also to train models on the data we expect to receive, as the user might not place their phone exactly horizontally and on the exact centre of the windshield. We added horizontal flips, random rotation up to 10 degrees, randomly zooming in on the image up to 20% and shifting vertically and horizontally the images to up to 10% of the image. However, this augmentations did not drastically increase the overall accuracy. As we were resource constrained we decided to train on resized images with height and width of 224 pixels. Images were therefore 4.5 and 9.1 times smaller in height and width respectively. This reduced the amount of memory and training time significantly.

$PD_{v2}^{MobileNetV2}$ had the same parameters as the $PD_{v1}^{MobileNetV2}$ however, we decided to train it for more epochs (40). This managed to increase accuracy from the 51% to 57% which is 7% better then the majority classifier (Table 1). We still had problems with the over-weighted class as the 'ignore' images were always classified as the majority class and the 'be careful' images were also classified as the majority class in 96% of cases (Table 2). The majority class (safe pedestrian) had a great recall of 99% and precision 55% (F1-score 71%). Brake class had a precision of 88% and recall 43% (F1-score 58%). In Figure 5 we can see that the accuracy on the valida-

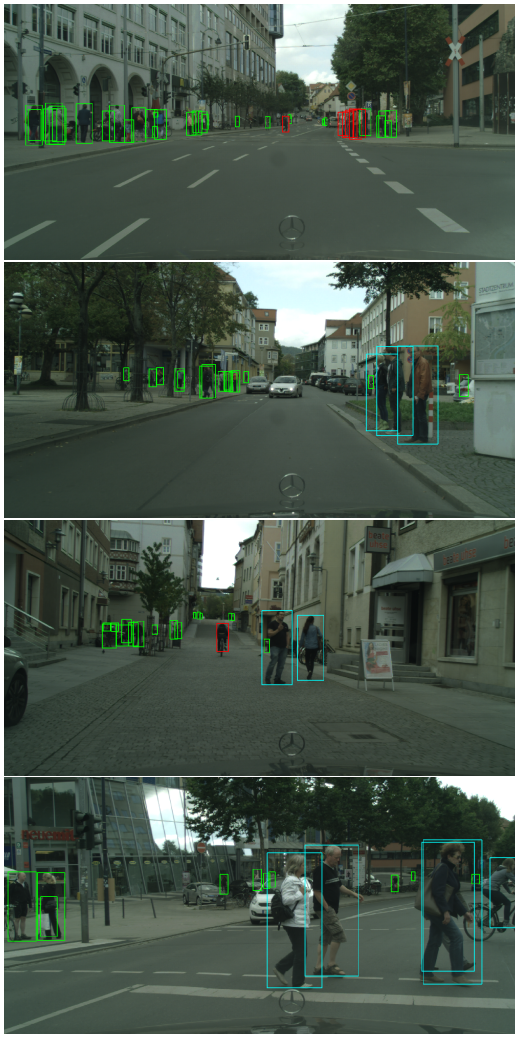

Figure 4: Examples from the dataset.

tion set did not increase largely after a few beginning epochs. The loss still looks like it is decreasing meaning probabilities for classes are getting better but the most probable class is not accurate.

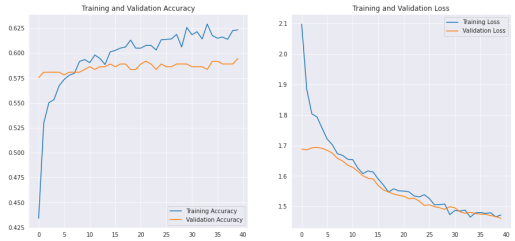

Figure 5: Accuracy and loss on train and validation set during training for $PD_{v2}^{MobileNetV2}$.

$PD_{v3}^{MobileNetV2}$ had a trainable base model. The training time increased approximately by 4 times comparing to the models with frozen weights of the MobileNetV2

	precision	recall	f1-score	support
Ignore	0.00	0.00	0.00	59
Safe pedestrian	0.55	0.99	0.71	252
Be careful	0.00	0.00	0.00	108
Brake	0.88	0.43	0.58	81
accuracy			0.57	500
macro avg	0.36	0.36	0.32	500
weighted avg	0.42	0.57	0.45	500

Table 1: Model scores for $PD_{v2}^{MobileNetV2}$.

	I	S	C	B
I	0	**59**	0	0
S	0	**250**	1	1
C	0	**104**	0	4
B	0	**43**	3	35

Table 2: Confusion matrix for $PD_{v2}^{MobileNetV2}$.

[8] base model. Despite the increased time, we observed the increase of accuracy by 2% (59%). In Figure 6 we can see that the model was only trained on 10 epochs because of the resource and time constraints. We can see that the validation accuracy would probably increase a bit further if we trained for more epochs. In Table 3 we can see the increase in F1-scores for all classes. The confusion matrix in Table 4 confirms that $PD_{v3}^{MobileNetV2}$ is a better model as the brake class gets identified in the majority of cases correctly and other classes that were always miss-classified get a few correct classifications.

Figure 6: Accuracy and loss on train and validation set during training for $PD_{v3}^{MobileNetV2}$.

	precision	recall	f1-score	support
Ignore	0.47	0.15	0.23	59
Safe pedestrian	0.57	0.94	0.71	252
Be careful	0.50	0.04	0.07	108
Brake	0.81	0.54	0.65	81
accuracy			0.59	500
macro avg	0.59	0.42	0.42	500
weighted avg	0.58	0.59	0.51	500

Table 3: Model scores for $PD_{v3}^{MobileNetV2}$.

	I	S	C	B
I	9	**50**	0	0
S	7	**238**	3	4
C	3	**95**	4	6
B	0	36	1	**44**

Table 4: Confusion matrix $PD_{v3}^{MobileNetV2}$.

For the $PD_{v4}^{MobileNetV2}$ we decided to change the input size of the images from the 224x224 to 512x512. We also decreased the batch size from 8 to 4. Because of the increased input the training of 1 epoch took around 7 times more time than for $PD_{v3}^{MobileNetV2}$ and around 28 times more than $PD_{v2}^{MobileNetV2}$. This is why we were forced to decrease the training to only 5 epochs. However, even with such short training we were able to increase the accuracy by another 2% to 61% (Table 5). In Table 6 we can see that the model increased the number of correctly classified 'be careful' and 'brake' classes which are the most important for our application as they are the ones where we have to alert the user.

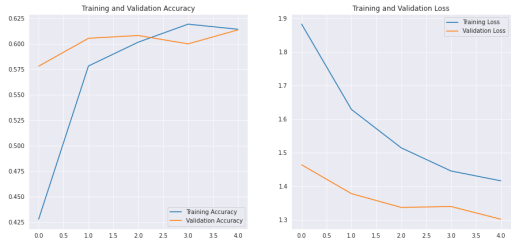

Figure 7: Accuracy and loss on train and validation set during training for $PD_{v4}^{MobileNetV2}$.

	precision	recall	f1-score	support
Ignore	0.00	0.00	0.00	59
Safe pedestrian	0.60	0.94	0.73	252
Be careful	0.50	0.23	0.32	108
Brake	0.85	0.54	0.66	81
accuracy			0.61	500
macro avg	0.49	0.43	0.43	500
weighted avg	0.55	0.61	0.54	500

Table 5: Model scores for $PD_{v4}^{MobileNetV2}$.

	I	S	C	B
I	0	**59**	0	0
S	0	**238**	11	3
C	0	**78**	25	5
B	0	23	14	**44**

Table 6: Confusion matrix for $PD_{v4}^{MobileNetV2}$.

We also tried different architectures of the base model. The $PD_{v1}^{EfficientNetB0}$ reached 48% accuracy. The $PD_{v1}^{DenseNet12}$ did reach a decent accuracy of 62%. However, its size is more than twice bigger than the MobileNetV2 and it reaches really similar accuracy.

The main difference of $PD_{v5}^{MobileNetV2}$ to the previous model is using a different distribution of the dataset as discussed previously in the Data section. Because of this we managed to increase the accuracy of the model by 5% to 66%.

	precision	recall	f1-score	support
Ignore	1.00	0.01	0.03	77
Safe pedestrian	0.69	0.88	0.77	287
Be careful	0.54	0.49	0.51	90
Brake	0.68	0.74	0.71	46
accuracy			0.66	500
macro avg	0.73	0.53	0.50	500
weighted avg	0.71	0.66	0.60	500

Table 7: Model scores $PD_{v5}^{MobileNetV2}$.

	I	S	C	B
I	1	**74**	2	0
S	0	**252**	28	7
C	0	37	**44**	9
B	0	4	8	**34**

Table 8: Confusion matrix for $PD_{v5}^{MobileNetV2}$.

3.3 Android app

Our Android application that utilises the model discussed previously consists of a few important parts. The user interface used to interact with the user. The camera used to capture images. The machine learning model that takes the image from the camera and returns the prediction. The last important part is the sound which plays if the 'brake' prediction appears. When the user first opens the app, they are greeted by a screen that shows three buttons. The first is called 'detect' and starts image capture from the camera. The second is called 'settings' and allows the user to adjust the settings. And the third is called 'exit' and exits the application.

When the user clicks detect the user is redirected to a new activity which starts the image capture from the phone's camera. After the image is captured it is fed to the machine learning model which predicts one of the four labels: 'ignore', 'safe pedestrian', 'be careful' or 'brake'. After the prediction is obtained the application checks if the brake prediction is present and if it is, it starts the handler which handles the playing of the beeping sound every three seconds until the brake prediction disappears. The important thing to note here is

that if the handler was already initialized it is not triggered again meaning we don't have multiple handlers playing the beeping sound at once. The phone screen when the application is detecting pedestrians is shown in Figure 8

Figure 8: Main screen of our application.

3.4 Experiments

Our application was tested in various different scenarios. It was first tested in a closed environment with the video playing in front of the phone. The video that was played was a drive in downtown of Los Angeles. This was done to simulate driving in a city where there is a lot of opportunities that a brake prediction may arise. The next way our application was tested was on a drive in the real world from Ljubljana to Delnice to see how our application performs in different real world environments. The chosen road is nice as it contains a lot of different environments like city, fields and forests. In the real world testing the phone was placed about 30 centimetres away from the windshield of the car. During the real world drive there were a lot of different weather conditions as well like cloudy, rainy and fogy.

4. RESULTS & DISCUSSION

The performance of our application when tested with the video is mainly good as it doesn't beep if there is no danger when pedestrians are present and it only beeps when the pedestrians are in front of the car. One of the things that can be seen in the video is that the application beeps if the car is stationary which is not that good, but is somewhat bearable by the beeping being played only every 3 seconds. One other thing to note with the video is that our application also beeped when the car was turning in the corner which can be a problem.

When we tested our application in the real world on a drive from Ljubljana to Delnice there was barely any brake situation present which can be understandable because we tested it on a sunday and there are not many people on the street on sundays. Also when testing in the real world we noticed that our application did not beep when there were no pedestrians present, when driving in the woods for example. However, we did notice that our application in those situations did not predict the 'ignore' label, but rather it predicted the 'safe pedestrian' label or no label at all. An important thing we also kept an eye out for on our real world testing was the battery consumption and we noticed that on the drive from Ljubljana to Delnice with the phone running our application with the mobile data turned off, the phone's battery dropped from 97% to 37% which is a battery loss of 60% which is quite a lot. An important thing to note here is that this was tested on the Samsung Galaxy S22 Plus model phone.

As we can see in the confusion matrices, the models even after the weight balancing of the classes prefer to predict the safe pedestrian class. This could be fixed by training the models for more epochs, as the models did not reach the overfitting stage. We would also benefit if the classification was binary for example the safe class and the brake class. However, we did find that the accuracy of the models increases if we increase the threshold of the confidence that is needed to predict a label. For example when we increased the threshold to 50% confidence the models accuracy was 67% while it predicted 90% of the input images. The model accuracy increased to 69% when we increased the confidence to 60%, but we only predicted 77% of the input images. The distribution of labels that were not predicted was similar to the distribution in the test set. We also found out that after the increase of threshold to 50% the models produced labels that were more stable, meaning that the predicted label does not change from frame to frame but from situation to situation.

4.1 User feedback

As a part of our testing and evaluation, we also gave the video of our application predicting on a city drive to other users to tell us a few words and improvements we can do with our application. One of the users said that the sound should be longer to account for the driver's physical state like tiredness. The same user also said that it works well if the speed is constant, but if the speed was not constant than the beeping would not be in time and the user would not be able to stop. Another user said that they did not like the sound because it sounded more like a danger and less like a warning. They also said that when the car was turning the application beeped to late. The third user said that distance from the car along with speed should be considered when playing the beeping sound. This user also said that the application shouldn't beep if the person is moving away from the car. Other than the remarks mentioned above the users liked our application.

5. CONCLUSION & FUTURE WORK

In this paper we presented our solution for safer roads. We presented the computer vision models that we developed and integrated in a working application. The ap-

plication does not reach a high accuracy, however once we increased the confidence threshold the app works satisfactory. Nevertheless, the application needs further training and testing in various road conditions and different geographical places, as the road conditions can vary extremely. We also believe that we should train our models as a binary classifiers, because the purpose of the application is to alert the driver to dangerous situations. In the future we should also increase the battery time by allowing our application to run in the background, by using different FPS, etc. We should also consider different CV models that might produce the same accuracy at a lower resource cost. Increasing the input image increased the accuracy of the model by few percent, we might consider inputting a larger image into our models. We should also consider the speed of the car, as the braking path increases with speed.

Author Contributions

Aljoša obtained the CityPerson dataset, created a Python script to manually annotate it with additional labels and annotated it. Aljoša did the data exploration and prepared data for training and testing. He trained the models and compared the results between different models and hyper parameters. Aljoša also did some small fixes on the Android app.

Žan created the Android app, programmed the sound alert. He integrated computer vision models in the app. He performed real world testing and helped with annotating the data.

We both contributed to the presentations, report and user evaluation.

6. REFERENCES

[1] "Transportation Safety: Pedestrian safety." https://www.cdc.gov/transportationsafety/pedestrian$_safety/index.html.Accessed$: $2022 - 10 - 14$.

[2] F. García, J. García, A. Ponz, A. de la Escalera, and J. M. Armingol, "Context aided pedestrian detection for danger estimation based on laser scanner and computer vision," *Expert Systems with Applications*, vol. 41, no. 15, pp. 6646–6661, 2014.

[3] "Pedestrian detection from a moving vehicle," *Proc. of the European Conference on Computer Vision*.

[4] "A trainable pedestrian detection system," *Proceedings of Intelligent Vehicles*.

[5] S. Zhang, R. Benenson, and B. Schiele, "Citypersons: A diverse dataset for pedestrian detection," *CoRR*, vol. abs/1702.05693, 2017.

[6] M. Cordts, M. Omran, S. Ramos, T. Scharwächter, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset," in *CVPR Workshop on The Future of Datasets in Vision*, 2015.

[7] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[8] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," 2018.

[9] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009.