BASCAT KIWIS: SEGMENTATION OF CAR PARTS USING DEEP LEARNING

Aniol Bisquert Parés, Ander Barrio Campos, Eduard Puga Creus and Harjodh Singh

s231813, s231938, s231919, s231481

ABSTRACT

The current insurance claims process, especially in the assessment of vehicle damages, heavily relies on manual labor, resulting in a time-consuming, resource-intensive, and error-prone procedure. The accurate identification of car parts is crucial for streamlining insurance claims. In collaboration with Deloitte, this study proposes the integration of deep learning techniques to develop an image segmentation solution, ensuring the precise identification of car parts in insurance claims. The research introduces and compares two approaches: one involving the creation of a model from scratch following the U-Net structure, and the other utilizing a pre-trained DenseNet121 model. The primary objective is to showcase the efficacy of these models in automating insurance claims, thereby facilitating a transformative shift from manual to automated processes.

Index Terms— Deep Learning, Convolutional Networks, Semantic Segmentation,

1. INTRODUCTION

Currently insurance claims, the process of assessing damages to vehicles is an inefficient one, which is heavily reliant on manual labour. When analysing images to segment them and evaluate the extent of damage to each car parts. This approach not only consumes significant time and resources but also has the possibility of human error. As a result, a collaborative project between Deloitte Consulting was undertaken with the ambition to implement an automated car image segmentation process for insurance claims.

This project is intended to use deep learning techniques and methodologies to develop image segmentation models which are capable of identifying and distinguishing car parts. Image segmentation is defined as the procedure of separating an image into multiple segments and can be used to locate objects and boundaries of images. In the future, these models could potentially be used to aid with insurance claims for not only cars but other motor vehicles and other assets.

One of the main points of this collaborative effort was around the development of data manipulation techniques, augmentation, and preprocessing, together with the creation of an image segmentation model powered by deep learning methodologies. The goal was clear: achieve better results than the manual assessment and introduce a shift towards automation. consequently diminishing the time required for processing insurance claims while enhancing accuracy, reliability, and efficiency.

Upon investigation into uses of Deep Learning for image segmentation, it is found that it has primarily been used for medical purposes. It has become a powerful technique in segmenting different parts of CT scans[], MI Scans []. Looking at the specific methods used in medical paper, a plethora of deep learning architectures were used such as; U-Net, SegNet, Fully Convolutional Methods and DenseNets. The models used in this report are U-Net and DenseNet121.

This report encapsulates the details used in the processes through this collaboration, presenting a comprehensive analysis of the models employed, the outcomes achieved, and the recommendations born from the results. The aim is to showcase the model performance and its viability in transforming insurance claims from a manual to an automated process.

2. DATA SET

The initial data set provided courtesy of Deloitte, consisted of four main categories; black 5 door, orange 3 door, car photos and landscape. The car images are each then separated into further subcategories of segmented and non-segmented. Where the segmented image contains a car image with the segmented parts manually highlighted, and the non-segmented is the original photo. Furthermore, the number of items in each category are very unevenly distributed, which is addressed in image preprocessing and augmentation.

Category	Number of items
black_5_doors	800
orange_3_doors	2000
car photos	170

Table 1: Number of Items in Dataset

Fig. 1: Data Set Examples

3. PREPROCESSING AND AUGMENTATION

All images in the initial dataset were standardized to a uniform size of 224 x 224. Additionally, images in PNG format were converted into arrays using a function designed to transform RGB-colored masks into numeric labels based on a provided mapping table. Subsequent to this preprocessing, data augmentation techniques were applied to address class imbalances and enhance model generalization. Categories with underrepresented photos, such as the "photo" category, underwent 11-fold augmentation, while overrepresented categories like "orange" or "black_5_doors" were augmented once and twice, respectively. Augmentation techniques included rotation for variations in orientation, zooming to magnify or reduce specific regions, and horizontal flipping to further diversify the dataset. In total, the initial dataset of approximately 3000 car images was transformed into the "DeloitteAug" dataset, containing around 11000 car images, prepared for training the model in the car parts segmentation task.

4. MODELS

In this section, we look at two ways to make models: starting from scratch and using pre-trained ones. For our task, we're comparing models we built from the ground up, with all the details we know, and models that already learned stuff beforehand. We want to see what works best and share what we find.

4.1. Car Parts Segmentation with U-Net

In the sector of image segmentation, the U-Net architecture has been accepted as a one of the leadings models [1] , especially noted for its effectiveness in biomedical contexts. Consequently, a model for the segmentation of car parts has been developed from scratch, leveraging the strengths of the U-Net architecture.

The primary goal of the U-Net model in this project was predicting segmentation masks, identifying the category that each pixel belongs to in terms of car parts. This task involves detailed pixel-level classification.

The U-Net model architecture implemented in this project follows a standard convolutional network structure optimized for segmentation tasks. The architecture have different convolutional blocks with batch normalization and ReLU activation, followed by max pooling and dropout layers for regularization. The encoding path consists of convolutional blocks with 64, 128, 256, and 512 filters, respectively, each followed by pooling and dropout. The bottleneck layer uses 1024 filters and finally the decoding path mirrors the encoding path with upsampling, concatenation, and convolutional blocks.

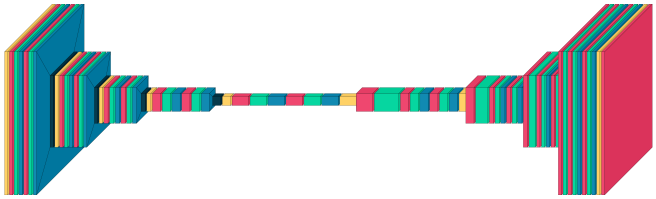

Fig. 2: U-Net architecture

4.1.1. U-Net Training Optimizations

In this section, the optimizations and improvements implemented in the training of the U-Net model for car parts segmentation are presented.

- **Data Generator:** Data generator were utilized in this project in order to efficiently manage the images in small batches and don't run out of RAM memory in the process.
- **Early Stopping and Model Checkpoint:** Early stopping was implemented during the model's training in order to stop the training if the model is no longer learning, thereby preventing overfitting. Additionally, model checkpoint was used in order to save the best effectives version of the model, determined by its performance on the validation dataset.

- **GPUs:** For the training of the model, two T4 GPUs were utilized. Initially, the model was configured to train on a single GPU, but appropriate code modifications were made to leverage both GPUs simultaneously, significantly enhancing the training efficiency. This adaptation reduced dramatically the training process and allow us to train more different hyperparameters configurations.

4.1.2. U-Net Training Procedure

Hardware The training was performed in Kaggle notebooks, using one CPU with 30GB of RAM and two T4 GPUs with 15GB of RAM each one.

Training Process and Batching The training process was performed using batches of 16 images, an optimal size that balanced the computational load and memory usage effectively, for a total of 50 epochs, using early stopping as a regularization technique and Adam as optimizer with a learning rate of 1×10^{-4}. The learning process ran for 5 hours. For this model, the selected loss function is the Dice loss, as detailed in the subsequent section.

4.2. Car Parts Segmentation with DenseNet121

The semantic segmentation model also relies on a pre-trained and fine-tuned convolutional neural network (CNN). The chosen arquitecture has been the DenseNet121 [2], due to it's high effectiveness in image classification. We obtained the pre-trained DenseNet121 model from TensorFlow's Keras applications library and it comes with weights learned from the well known ImageNet dataset.

DenseNet121 it's acting as the feature extractor. We decided to exclude the densely connected fully connected layers by setting *include_top* to *False* to retain only the convolutional backbone. This allows us to use hierarchical feature hierarchical representations while adapting the architecture to our specific task. As one part of maintaining the integrity of learned features, we have decided to freeze all layers of the DenseNet121 backbone to fine-tune subsequent layers for optimal performance.

The decoder then augments the model with a series of convolutional and upsampling layers, progressively expanding spatial dimensions. What sets our decoder apart is its ability to merge information from intermediate layers of the DenseNet121 backbone. Specifically, we incorporated concatenation points such as *conv4_block24_concat*. The combination layers help our model mix tiny details with the bigger picture, creating a clear understanding of the information we are looking at.

The model concludes with a convolutional layer that uses softmax activation, creating different probability maps for each of the nine classes in the segmentation task. What this layer is doing is to transform feature representations from the decoder into accurate predictions across the specified classes.

To sum up, our model combines the feature extraction power of DenseNet121 with a customized decoder for semantinc segmentation. We use the pre-learned skills from DenseNet121 and adjust them to fit what we need. Our goal is to achieve robust and accurate semantic segmentation [3] results in our car pictures.

4.2.1. DenseNet121 Training Optimizations

It has to be mentioned that DenseNet121 is a pre-trained model. However, it has been fine-tuned and the training optimizations implemented for the new model follow the same structure regarding to the Data Generator, Early Stopping and Model Checkpoint and GPU used in the U-Net.

4.2.2. DenseNet121 Training Procedure

Once again, for the fine-tuned model, the training process was performed using the same exact values and parameters as the U-Net for the batches (16), number of epochs (50), early stopping and optimizer. The only difference here has been the type of loss used. In this case, the Categorical Cross Entropy loss has been used. The decision to use this loss aligns with the project task's characteristics and has proven effectiveness, promoting efficient training convergence and precise semantic segmentation in input images

5. EXPERIMENTAL RESULTS AND COMPARISONS

5.1. Metrics

Accuracy is the ratio of properly classified pixels divided by the total number of pixels. For $K+1$ classes.

$$\text{Accuracy} = \frac{\sum_{i=0}^{K} p_{ii}}{\sum_{i=0}^{K} \sum_{j=0}^{K} p_{ij}} \tag{1}$$

where p_{ij} is the number of pixels of class i predicted as belonging to class j.

Mean IoU is defined as the area of intersection between the predicted segmentation map A and the ground truth map B, divided by the area of the union between the two maps, and ranges between 0 and 1.

$$\text{IoU} = \frac{|A \cap B|}{|A \cup B|} \tag{2}$$

Mean-IoU is defined as the average IoU over all classes.

Dice Coefficient Commonly used in medical image analysis, can be defined as twice the overlap area of the predicted and ground-truth maps divided by the total number of pixels.

$$\text{Dice coef} = \frac{2|A \cap B|}{|A| + |B|} \tag{3}$$

5.2. Data Splitting

The dataset was splitted into training, validation, and test sets. A significant portion, 80%, of the data was used for training in order for the model to learn the different patterns in the data, a 20% of the data was reserved for validation to provide a dataset for comparing the model's performance. Finally, for the test set, 30 specific images provided by the Deloitte team were used. This division was crucial in evaluating the model's ability to generalize across different data samples.

5.3. Loss

Categorical Cross Entropy loss For a multi-class segmentation task the most popular loss is the categorical cross entropy loss (cce) which each image pixel is labeled with a category number and a model is trained to predict that number for each pixel. The formula for Categorical Cross-Entropy loss is as follows:

$$L(y, \hat{y}) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{K} y_{ij} \cdot \log(\hat{y}_{ij}) \tag{4}$$

Where: - N is the number of samples in the dataset. - K is the number of classes. - y_{ij} is a binary indicator of whether class j is the correct classification for sample i. - $\hat{y}_{ij}$ is the predicted probability that sample i belongs to class j.

Dice loss is an effective choice for cases where the classes are unbalanced as introduced by Sudre et al. [4]. The formula for Dice Loss is as follows:

$$\text{Dice Loss} = 1 - \text{Dice coef} \tag{5}$$

5.4. U-Net on DeloitteAug dataset

The training procedure took more than 5 hours. As seen in Figure 3, the validation loss did not show an improvement after epoch 38 remaining at 0.0213. As shown, there are no signs of overfitting during the training process thanks to the well balanced data done during the data augmentation phase of the project.

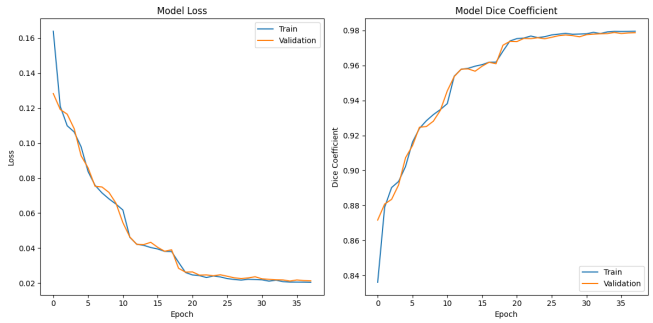

Fig. 3: U-Net learning curves

The performance metrics of the model across training, validation, and test datasets are summarized in Table 2. In the training phase, the model achieved a loss of 0.0198, an accuracy of 98.03%, a mean IoU of 97.80%, and a Dice coefficient of 98.03%. Validation results were consistent with a loss of 0.0212, an accuracy of 97.89%, a mean IoU of 97.65%, and a Dice coefficient of 97.88%. During testing, the model demonstrated robust performance with a loss of 0.0265, an accuracy of 97.36%, a mean IoU of 96.91%, and a Dice coefficient of 97.35%. The summarized results provide a comprehensive overview of the model's evaluation metrics, showcasing its effectiveness and generalization across diverse datasets.

	Dice Loss	**Dice Coef**	**MeanIoU**	**Accuracy**
train	0.0198	98.03%	97.80%	98.03%
validation	0.0212	97.88%	97.65%	97.89%
test	0.0265	97.35%	96.91%	97.36%

Table 2: U-Net Results

Figure 4 compares U-Net predictions with ground truth masks on some images from the test set. As it can be observed, the model demonstrates strong performance in segmenting various car parts in real, unseen test images. However, it faces minor challenges in accurately segmenting the door handle (a)), certain rear lights (b)), and the car brand logo (c)). Overall, the model exhibits robustness in capturing intricate details, and despite minor limitations, it proves effective in handling complex visual information in image segmentation tasks.

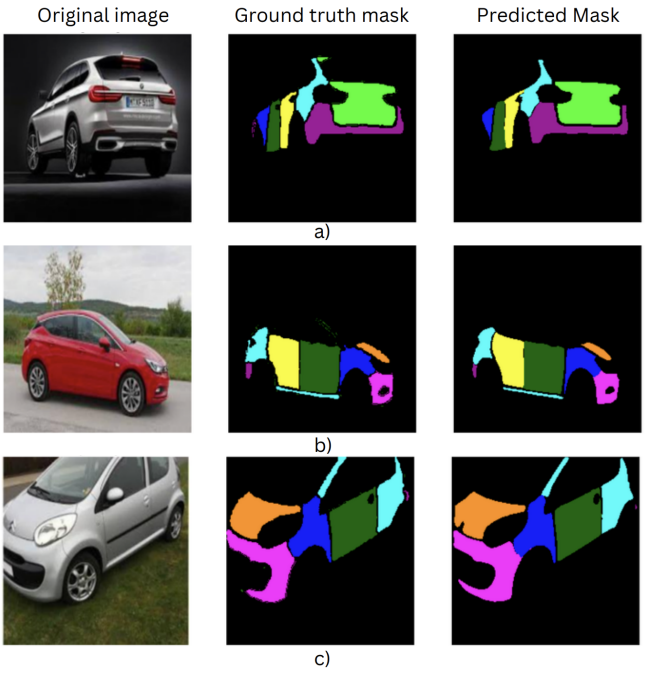

Fig. 4: U-Net performance

5.5. DenseNet121 on DeloitteAug dataset

Pretrained DenseNet121 model was faster to train compared to the U-Net one. As seen on Figure 5, the cce loss did not decrease anymore after around 20 minutes and stopped at epoch 26 remaining at 0.1087.

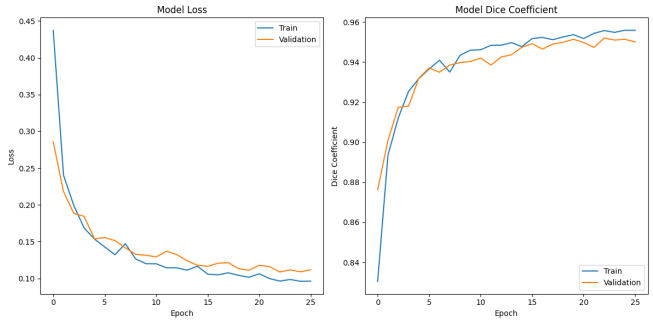

Fig. 5: DenseNet121 learning curves

The performance metrics for the DenseNet121 model across training, validation, and test datasets are summarized in Table 3. During the training phase, the model exhibited excellent performance, achieving a cce loss of 0.0947, an accuracy of 96.95%, a mean IoU of 79.62%, and a Dice coefficient of 95.66%. The validation results remained consistent, with a loss of 0.1102, an accuracy of 96.51%, a mean IoU of 79.00%, and a Dice coefficient of 95.13%. In the testing phase, the model demonstrated robust performance, yielding a loss of 0.1500, an accuracy of 95.32%, a mean IoU of 68.90%, and a Dice coefficient of 93.21%. The detailed evaluation metrics for the training, validation, and test datasets are provided in the corresponding sections. These results underscore the effectiveness and generalization of the DenseNet121 model across diverse datasets, emphasizing its potential for accurate car parts segmentation.

	cce Loss	Dice Coef	MeanIoU	Accuracy
train	0.0947	95.66%	79.62%	96.95%
validation	0.1102	95.13%	79.00%	96.51%
test	0.1500	93.21%	68.90%	95.32%

Table 3: DenseNet121 Results

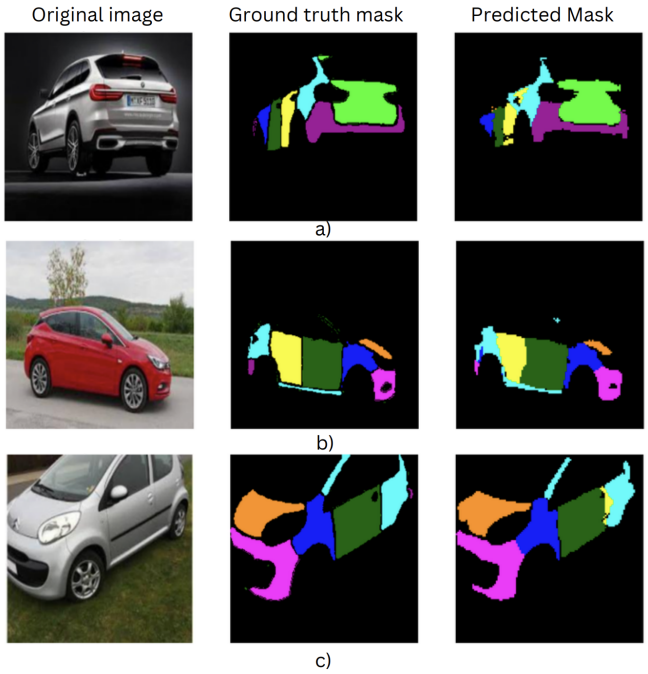

Fig. 6: DenseNet121 Performance

The visual representation in Figure 6 illustrates the results of our model's performance on segmented car parts using images from the test set. The predicted masks, which represent the segmented areas of the cars, closely resemble the ground truth masks. However, upon closer inspection, it becomes evident that the segmented parts exhibit a higher level of pixelation compared to their original counterparts.

Additionally, there are instances of misclassification where certain car parts are incorrectly identified. For example, in the case of the car depicted in b), the model mistakenly classifies the rear quarter panel (purple) as the bumper (pink). Another misclassification occurs in the example of the car in c), where the model segments a small second door frame (yellow) for a vehicle that actually has a unique door.

6. DISCUSSION

This study explored the use of U-Net and DenseNet121 to segment car parts, aiming to simplify the evaluation of vehicles in insurance claim processes. Although both methods performed well, there were noticeable differences between them.

U-Net showed slightly better results than DenseNet121 across all performance metrics, especially in MeanIoU. This could be attributed to U-Net's use of Dice Loss during training, which optimizes MeanIoU, a metric closely related to the Dice Coefficient. The Dice coefficients for both models suggested effective generalization without signs of overfitting.

When visually assessing the predicted segmentation masks, U-Net achieved a significant three-point higher dice

score in the test split. It consistently and accurately segmented all car parts, providing clearer boundaries compared to DenseNet121. This success may be due to U-Net's weights being trained exclusively on the task-specific "DeloiteAug" dataset. In contrast, the pretrained weights of DenseNet121 might perform better for a different distribution of car images.

However, it's important to note that the U-Net model required considerably more time for training compared to the pretrained DenseNet121 model. This becomes crucial in situations with limited resources, as observed during the development of this study. Balancing superior performance with computational efficiency should be carefully considered in practical applications, particularly when training time is a critical factor.

Significant limitations regarding RAM memory were found when Google Colab was used for computational needs in this project. These limitations are due to the need to handle large datasets and the complexity of deep learning models, which require substantial computational resources. Consequently, a transition was made to Kaggle notebooks, which provided a higher RAM capacity.

However, even the increased RAM offered by Kaggle notebooks proved insufficient for the project's demands, especially due to data augmentation and the processing requirements of the U-Net and DenseNet121 models. Therefore, data generators were utilized for batch loading of data, facilitating the efficient management of large volumes of images while avoiding the exhaustion of available RAM, by loading data in batches, optimal performance was maintained without compromising the training process's integrity.

In the preprocessing step of transforming the provided images into mask maps, an additional limitation was encountered. The colors in the images did not map precisely to the expected mask representations and to address this discrepancy, the establishment of a threshold was required to enable accurate segmentation of the different parts of the car in the images.

The encounter with this color mapping challenge points out the importance of meticulous preprocessing in image segmentation tasks, particularly in projects where color accuracy is vital. Future work could be the exploration of more sophisticated color mapping techniques or machine learning algorithms capable of dynamically adjusting thresholds based on image characteristics.

7. GITHUB REPOSITORY

In this github repository link can be found the corresponding notebooks: Car segmentation project

8. ACKNOWLEDGEMENTS

The authors wish to thank Martin Closter Jespersen, Asbjørn Eller Skaarup, Mikkel Sikker Sørensen, Nicklas Lund, Georgios Sartzetakis of Deloitte for providing not only their guidance and insights but also use of their internal dataset. In addition, we want to thank Ole Winther & Jes Frellsen for organising this collaborative opportunity with the Deloitte team.

9. REFERENCES

[1] Olaf Ronneberger, Philipp Fischer, and Thomas Brox, "U-net: Convolutional networks for biomedical image segmentation," 2015.

[2] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger, "Densely connected convolutional networks," 2018.

[3] Reza Zare and Arash Pourkazemi, "Densenet approach to segmentation and classification of dermatoscopic skin lesions images," 2021.

[4] Carole H. Sudre, Wenqi Li, Tom Vercauteren, Sebastien Ourselin, and M. Jorge Cardoso, *Generalised Dice Overlap as a Deep Learning Loss Function for Highly Unbalanced Segmentations*, p. 240–248, Springer International Publishing, 2017.