Computer Vision Automobile Classification Using Various Deep Convolutional Neural Network Architectures

Christensen M. Frans
Computer Science Department
Faculty of Computing and Media
Bina Nusantara University
Jakarta, Indonesia 11480
christensen.frans@binus.ac.id

Nunung N. Qomariyah
Computer Science Department
Faculty of Computing and Media
Bina Nusantara University
Jakarta, Indonesia 11480
nunung.qomariyah@binus.ac.id

Abstract* — This research entails the general & technical overview of modern Machine Learning algorithms, as well as how it is able to potentially benefit human beings by assisting a real-world problem; parking congestion. Specifically, its main objective is to prove that Computer Vision is able to classify different automobile types, in order to provide automation & replace the manual labor in categorizing cars, which is essential for a more efficiently designed parking layout. This project incorporates multiclass image classification using a Custom-Built, and multiple popular Deep Convolutional Neural Network architectures; the ResNet-50, Inception V3, Xception, DenseNet 201, and another DenseNet 201 Pre-Trained with ImageNet weights (Transfer Learning). Upon evaluation & comparison amongst the experimented architectures, the DenseNet 201 model is able to achieve the best performance in terms of Accuracy and F1 Scores of 76% and 0.71 respectively. Additionally, this research also concludes that removing backgrounds from the dataset images, or in other words excluding unnecessary noise from the input features, does have a significant positive impact on every models' performances & training times. On average, the Accuracy and F1 Scores surge by 14% and 16% respectively. Furthermore, the Execution Times have decreased by a range of 24 - 44% depending on the architecture. The outcomes yielded from the conducted experiments proves that Deep Learning is successful in achieving this research's main objective, thus showing that Computer Vision could indirectly contribute to decreasing parking congestion.

Keywords
Computer Vision, Deep Learning, Deep Convolutional Neural Network, Transfer Learning

I. Introduction

Automotive drivers experienced trouble looking for parking spots at least once in their lifetimes. According to USA Today, drivers spend on average about 17 hours every year just to look for parking spots. [1] This results in additional costs for both the driver and environment aside from just time alone; more fuel consumed & purchased, emissions, traffic congestions, and many more.

II. Problem

To prevent parking congestion, an efficient alternative would be to separate parking spots for smaller and larger cars, similar to how motorbikes and cars are typically segregated. This way, smaller cars will find more available lots; such as lower ceiling areas, shorter available distances from curbs, tighter backout path widths, & narrower corners. On the other hand, the saved spaces could be used for larger cars to park. Therefore, the final layout leaves more available parking spaces for all.

Unfortunately, this method requires a manual process in directing all vehicles that are entering the building, leading to additional costs of hiring staff. It is also impractical to completely rely on employees, as it's an exhausting yet monotonous task to standby & tell drivers where to park for hours straight; potentially causing human errors. Alternatively, a sign could be used so drivers can determine where to park by themselves. However, some drivers are not familiar with automobiles, thus this will rather confuse them, or worse cause them to park in the wrong section.

III. Solution

Fortunately, the procedure mentioned in the previous section could be automated with the help of Computer Vision in classifying different car body types, upon the car arriving at the ticket booth prior to entering the building. Instead of only reading car plate numbers and handing out parking tickets, the OCR camera in the entrance booth could also be used to capture the car's image and pass it into a trained Deep Learning model which predicts its category. The predicted output could then be printed out on the parking ticket handed out to the driver, which they can use to locate the dedicated parking section where their car should belong.

A. Project Scope

In order to incorporate Computer Vision to assist such operation, it is crucial to first prove that it is able to visually classify different automobile types. Ideally, more than one Deep Learning algorithm shall be considered for prototyping & experimentation. Hence, this research entails the performance comparison between 6 different Deep Convolutional Neural Networks (DCNNs) in performing multiclass automobile types classification; including 5 base and 1 Pre-Trained (Transfer Learning) models:

1. ResNet50
2. Inception V3
3. DenseNet 201
4. DenseNet 201 (Pre-Trained Imagenet Weights)
5. Xception
6. Custom-built DCNN Architecture

B. Project Aims

This research aims to achieve the following:

1. Build & evaluate which amongst the DCNN architectures experimented in this research is able to achieve the best classification results possible, or in other words the best performing model.
2. Train & predict the models on 2 types of datasets; with & without background removed. Thus, evaluate how removing backgrounds of the image data has impact on each of the model's performances

IV. Related Works

In October 2020, a paper titled "Real-Time Vehicle Classification" was published to IEEE (Institute of Electrical and Electronics Engineers) by 3 authors from Bangladesh. The research was aimed to reduce the number of road accidents in their country by using DCNNs to classify 4 of the most common types of cars in real time. The project came with multiple challenges, ranging from variation of shapes and colors in the image dataset. Fortunately, the final output reveals a 97% accuracy, which concluded a decent performance on the real-time test dataset. [2]

Another research published in February of 2021 titled "Convolutional Neural Network Based Vehicle Classification in Adverse Illuminous Conditions for Intelligent Transportation Systems" came with an objective to innovate the effectiveness of the current traffic control and highway automation systems. The authors claimed that the existing solutions were only trained on very limited and small datasets, and thus aren't able to cater real-time road traffic conditions. Alternatively, Deep Learning is incorporated to solve the above matters. In addition, the 10,000 image dataset, containing 6 different categories of vehicle types, that were used in training the models are mixed with random noises, such as weather conditions and illumination factors to improve its robustness in real-time applications. The project then assesses the performances of pretrained AlexNet, GoogleNet, Inception-V3, VGG, and ResNet models that were fine tuned based on the artificial dataset. Next, the best performing model, which is the ResNet architecture, was improved by adding a new classification block on the network. Moreover, to ensure generalization, the authors also fine tuned the network on the public VeRi dataset engulfing 50,000 images, which have also been categorized into 6 different vehicle classes. Finally, the proposed vehicle classification method was evaluated, revealing a 99.68% Accuracy, 99.65% Precision, and 99.56% F1 Scores. [3]

A recent project published in 2022 titled "CNN-Based Classification for Highly Similar Vehicle Model Using Multi-Task Learning" aspires to improve the existing intelligent traffic law enforcement systems, by implementing vehicle make and model classification in scenarios where traffic violations had been committed but the license plate failed to be obtained. The main challenge in this research is the fact that there are vehicles of different make and model vehicles but are very similar in visual appearances. To overcome this issue, a fine-grained DCNN classifier with multi-task learning is incorporated in the paper. The proposed approach begins with extracting features from the input images using the VGG-16 architecture. The extracted features are then split into 2 separate branches for classification; one for vehicle make, and the other one for vehicle model. Finally, the performance of the proposed method was evaluated with the InaV-Dash dataset, containing images from various Indonesian cars with very similar visual appearances. This result yields an Accuracy Scores of 98.73% and 97.69%, for vehicle make and model classification respectively. [4]

V. Research Method

A. Data Source

This project will make use of the Stanford AI Cars Dataset for model training & evaluation. It consists of 16185 car pictures of multiple different types, separated into an approximately equal amount of train & test portions. Initially, these images are categorized into 196 different class folders named after the cars' *year-make-model*; for instance: "2012 Tesla Model S", or "2012 BMW M3 Coupe". [5]

Figure V.1: Stanford AI Cars Dataset Preview [5]

B. Data Organization

The goal of the experiments is to classify car body types instead of *year-make-models*, hence the dataset needs to be modified into such categories instead. Since the dataset contains images that are already organized into folders named as their *year-make-model* categories, the next job is to manually determine which body type each of these cars belong to, and finally segregate their images accordingly based on it. This transforms the whole dataset to only having the following 7 classes:

1. Sedan
2. Sport
3. SUV
4. MPV
5. Hatchback
6. Wagon
7. Truck

C. Data Preprocessing

The first preprocessing step was to remove the backgrounds for all the images in the dataset. This technique neglects the unnecessary noises in features, so that it could be seamlessly interpreted by models during training. Regardless, a duplicate of the unmodified dataset is kept, which will later be used to compare the architectures' performances working on the dataset with & without their backgrounds removed.

Figure V.2: Removing Dataset Image Backgrounds Preview

The next preprocessing step is to rescale the RGB color channel values to a range of between 0 and 1, by dividing them all with its maximum possible value, which is 255.

Then, since the dataset used for this project contains images of different pixel dimensions, it is essential to normalize these image sizes to make sure that the number of features are standardized for each and every input, prior to feeding them into the DCNN architectures. Upon broad consideration, all pictures in the dataset was rescaled to a target size of 100 × 100 pixels.

D. Dataset Statistics

Upon reorganizing & preprocessing the dataset, it results an imbalanced dataset with the statistics shown below:

Table V.1: Dataset Category Distribution

Category	Number of Images
SUV	2846
MPV	1076
Sedan	4154
Sport	4537
Hatchback	1558
Wagon	411
Truck	1603

E. Train, Test, & Validation Data Split

The original Stanford AI Cars dataset is split into training & test portions of 50% each, thereby having roughly 8000 images in each set. Intuitively, this may not be enough for training the models to get the most optimized results. Moreover, they will also require a validation subset which is not included. Therefore, the portions were recompiled, reorganized, and split into a training size of 80%, validation & test sizes of 10% each. This way, the DCNN architectures will have approximately 13000 & 1600 pictures for training &

validation respectively, and the leftover of about 1600 images that have never been seen by the models to evaluate their performance.

F. Model Construction, Training, & Selection

The next step is to construct 6 of the following selected DCNN architectures for experimentation:

1. ResNet50
2. Inception V3
3. DenseNet 201
4. DenseNet 201 (Pre-Trained Imagenet Weights)
5. Xception
6. Custom-built DCNN Architecture

Moving forward, they are trained & tuned on behalf of the training & validation datasets with 100 Epochs each. Next, predictions are generated and assessed for each model with regards to the test dataset, through various evaluation methodologies. Then, performances across different models will be compared with one another, and finally the best performing model will be concluded.

G. Model Evaluation Techniques

Since the project revolves around a multiclass classification problem, and that the dataset used is imbalanced, the comparison in performance amongst the different models will be assessed through the Classification Report, consisting of the following metrics:

1. Accuracy Score
2. F1 Score
3. Precision Score
4. Recall Score

Moreover, the following trends will also be recorded and thereby evaluated for the Training & Validation sets for each model:

1. Accuracy Score VS Epochs
2. Loss Value VS Epochs

Finally, efficiency is also a crucial consideration in development, as it estimates the amount of resources that needs to be allocated for the process. Hence, the time complexity will also be assessed for each architecture; by observing the Training Times per Epochs trends as well as the overall Execution Times.

VI. Results & Discussion

A. Classification Report

The tables below entail each respective model's Classification Reports; listing the Accuracy, F1, Precision, & Recall Scores for every category:

Table VI.1: Classification Report for ResNet-50 Architecture

	Precision	Recall	F1 Score	Accuracy
Hatchback	0.575221	0.416667	0.483271	
MPV	0.817204	0.703704	0.756219	
Sedan	0.697674	0.724638	0.7109	
Sport	0.778523	0.769912	0.774194	
SUV	0.756173	0.862676	0.805921	
Truck	0.803191	0.94375	0.867816	
Wagon	0.666667	0.277778	0.392157	
Overall	0.727808	0.671303	0.684354	0.742236

Table VI.2: Classification Report for Inception V3 Architecture

	Precision	Recall	F1 Score	Accuracy
Hatchback	0.526012	0.583333	0.553191	
MPV	0.584906	0.861111	0.696629	
Sedan	0.735376	0.637681	0.683053	
Sport	0.729831	0.860619	0.789848	
SUV	0.905213	0.672535	0.771717	
Truck	0.872611	0.85625	0.864353	
Wagon	0.5	0.25	0.333333	
Overall	0.693421	0.674504	0.670304	0.729193

Table VI.3: Classification Report for DenseNet 201 Architecture

	Precision	Recall	F1 Score	Accuracy
Hatchback	0.589404	0.570513	0.579805	
MPV	0.9	0.666667	0.765957	
Sedan	0.772472	0.664251	0.714286	
Sport	0.763265	0.827434	0.794055	
SUV	0.811075	0.876761	0.84264	
Truck	0.797872	0.9375	0.862069	
Wagon	0.421053	0.444444	0.432432	
Overall	0.722163	0.71251	0.713035	0.76087

Table VI.4: Classification Report for DenseNet 201 (Pre-Trained ImageNet) Architecture

	Precision	Recall	F1 Score	Accuracy
Hatchback	0.407407	0.423077	0.415094	
MPV	0.666667	0.685185	0.675799	
Sedan	0.586517	0.630435	0.607683	
Sport	0.72028	0.683628	0.701476	
SUV	0.683502	0.714789	0.698795	
Truck	0.84507	0.75	0.794702	
Wagon	0.291667	0.194444	0.233333	
Overall	0.600158	0.58308	0.589555	0.645963

Table VI.5: Classification Report for Xception Architecture

	Precision	Recall	F1 Score	Accuracy
Hatchback	0.640625	0.262821	0.372727	
MPV	0.796117	0.759259	0.777251	
Sedan	0.787425	0.635266	0.703209	
Sport	0.674457	0.893805	0.768792	
SUV	0.793333	0.838028	0.815068	
Truck	0.878788	0.90625	0.892308	
Wagon	0.355556	0.444444	0.395062	
Overall	0.703757	0.677125	0.674917	0.738509

Table VI.6: Classification Report for Custom-Built DCNN Architecture

	Precision	Recall	F1 Score	Accuracy
Hatchback	0.666667	0.371795	0.477366	
MPV	0.793814	0.712963	0.75122	
Sedan	0.622309	0.768116	0.687568	
Sport	0.787952	0.723451	0.754325	
SUV	0.778502	0.841549	0.808799	
Truck	0.814607	0.90625	0.857988	
Wagon	0.333333	0.138889	0.196078	
Overall	0.685312	0.637573	0.647621	0.726087

Based on the Classification Report tables above, it could be noticed that indeed every single DCNN architecture achieves different scores across the metrics, especially for different categories. Unfortunately, the Wagon class had always been achieving the lowest scores. Coincidentally, this is also the category that has the least number of pictures in the imbalanced dataset. Hence, it is the confirmed culprit. On the other hand, Trucks seem to constantly achieve the best records for F1, Precision, and Recall Scores. From a logical and visual perspective, this may be due to the fact that they are usually much larger than other vehicles, or that they almost always have a very distinguishable "extended rear trunk" layout.

Likewise, the best performing model could be concluded by finding the top scoring model for these metrics. Upon observation, the DenseNet 201 architecture takes the lead on behalf of the Accuracy, F1 and Recall Scores; having that of 76%, 0.71, and 0.71 respectively. Alternatively for the Precision Score, the ResNet-50 only beats the DenseNet 201 by a tiny value of 0.005. Hence from this conclusion, the DenseNet 201 is crowned to be the best performing model through the Classification Reports.

B. Accuracy & Loss Trends

The following charts depict the Accuracy & Loss Trends that were recorded during the training process:

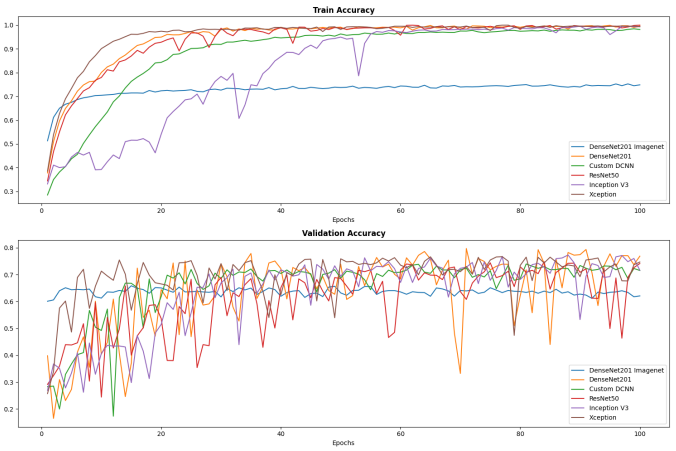

Figure VI.1: Accuracy Trend per Epoch for Each DCNN Architecture

The chart above conveys that every architecture's Accuracy levels similarly improve and eventually converge sideways as the number of Epochs increases during training. On the other hand, the validation subplot displays an unsteady outcome for nearly all the models. This is expected as they are validation scores, and unlike training scores.

If compared against one another, the ResNet-50, DenseNet 201, and Xception models raced for the top, while others like the DenseNet 201 (Pre-Trained ImageNet) are left behind. Also in the beginning, the Custom-Built DCNN architecture seems to accelerate in Accuracy level much faster than that of the Inception V3 model. Fortunately, the Inception V3 was able to catch up to it upon the 60th Epoch, and ended up performing even better after that. Alternatively, The DenseNet 201 did not converge to its most optimal Accuracy level as quickly as the Xception network, however, it still eventually ended up converging to a similar score after about 35 Epochs. On the validation chart, the DenseNet 201 may be a little more unstable than the competing Xception architecture, but it was still able to achieve higher than all the other models on many later iterations. Hence, this allows the DenseNet 201 architecture to take the lead for this comparison.

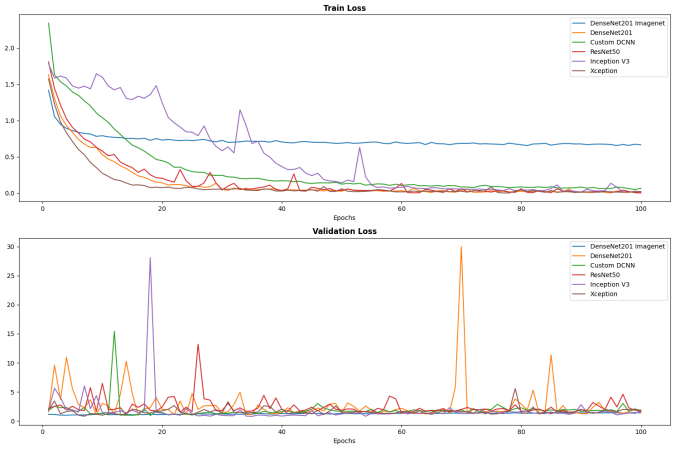

Figure VI.2: Loss Trend per Epoch for Each DCNN Architecture

Like the prior Accuracy charts, the Loss values per Epoch visualization above also presents similar but inverted trends for every DCNN architecture; overall converging downward and proving that the models were able to optimize themselves as the training iteration increases. For validation, it also follows an up-and-down trend like that of the Accuracy levels. Some architectures, such as the Custom-Built DCNN and Inception V3 have large spikes in the beginning iterations, while others like the DenseNet 201 have them on later stages.

Looking deeper and comparing the Loss trends amongst the DCNN architectures, it could be seen that again the ResNet-50, DenseNet 201, and Xception models are in tight competition to be having the least, while the DenseNet 201 (Pre-Trained ImageNet) converges at a higher Loss value. Also, it could be clearly depicted that the Xception model is able to outperform all the other models in terms of converging speed and optimization level. Regardless, like the previous comparison, both Xception and DenseNet 201 architectures converge to a pretty similar value. For the validation graph however, the Xception model clearly beats the DenseNet 201, alongside all the other models; with the least and less volatile Loss value. Thus, the triumph is awarded to the Xception network.

C. Training Time Efficiency

The following chart visualize the Training Times per Epoch for each of the DCNN architectures:

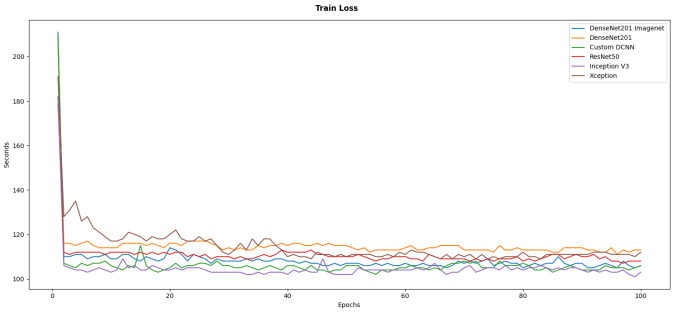

Figure VI.3: Training Time Trend per Epoch for Each DCNN Architecture

The graph above proves that different DCNN architectures built with different algorithmic ‘depths’, ‘widths’, and structures do indeed require contrasting training times. Overall, the training times seem to peak for all the models during the first training iteration, and eventually stabilize themselves right after. Regardless, nearly all the models maintained a training time range of between 100 and 120 seconds. When added up, the time it took to construct, train, predict, & evaluate each architecture are concluded in the total Execution Time table below:

Table VI.7: DCNN Architectures’ Execution Times

DCNN Architecture	Total Execution Time (Hours)
ResNet-50	2.96
Inception V3	2.96
DenseNet 201	3.26
DenseNet 201 (Pre-Trained ImageNet)	3.21
Custom-Built DCNN Architecture	2.73
Xception	3.25

It could be seen that this time, the Custom-Built DCNN takes the lead for the most time efficient results, followed by the Inception V3, ResNet-50, and DenseNet 201 (Pre-Trained ImageNet) architectures. Alternatively, the Xception and DenseNet 201 architectures seem to be having longer total Execution Times.

D. Best Performing Model Selection

Overall, the DenseNet 201 architecture seems to attain the most promising results, gaining traction from most of the preceding comparison methodologies but Execution Times. The Xception network may seem comparable from the Accuracy and Loss trends, but it scores 4% less in its overall Accuracy score than that of the DenseNet 201, which is in fact a pretty hefty amount. Upon consideration, the DenseNet 201 network shall maintain its crown, as intuitively the final output evaluation metric scores means more than its training processes. Lastly, although the DenseNet 201 comes off as the least Time Efficient model, it is merely just 30

minutes less, or equivalent to just below 20 percent slower, than the fastest Custom-Build DCNN architecture. Nonetheless, such tradeoffs cannot justify an 8 percent decrease in performance for the latter model. Hence, the DenseNet 201 architecture is elected as the best performing model in this research.

E. Impact of Removing Background - Classification Report

The following tables & figures concludes the change of the two most crucial metrics in the Classification Reports; the Accuracy & F1 Scores, when the dataset image backgrounds are maintained:

Table VI.7: Accuracy Scores Comparison (With & Without Background Removed)

Model	Accuracy Score - Background Maintained	Accuracy Score - Background Removed	Change
ResNet-50	0.620474	0.742236	12.1762
Inception V3	0.438202	0.729193	29.0991
DenseNet 201	0.686642	0.76087	7.4228
DenseNet 201 (Pre-Trained ImageNet)	0.543071	0.645963	10.2892
Custom-Built DCNN Architecture	0.564919	0.726087	16.1168
Xception	0.630462	0.738509	10.8047

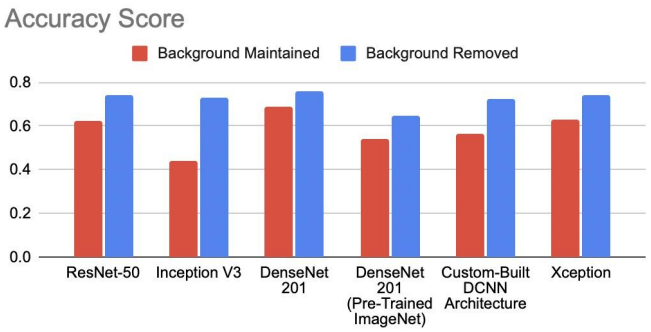

Figure VI.4: Accuracy Scores Comparison (With & Without Background Removed)

Table VI.8: F1 Scores Comparison (With & Without Background Removed)

Model	F1 Score - Background Maintained	F1 Score - Background Removed	Change
ResNet-50	0.543372	0.684354	14.0982
Inception V3	0.332846	0.670304	33.7458
DenseNet 201	0.631264	0.713035	8.1771
DenseNet 201 (Pre-Trained ImageNet)	0.478175	0.589555	11.138
Custom-Built DCNN	0.465975	0.647621	18.1646
Architecture			
Xception	0.569577	0.674917	10.534

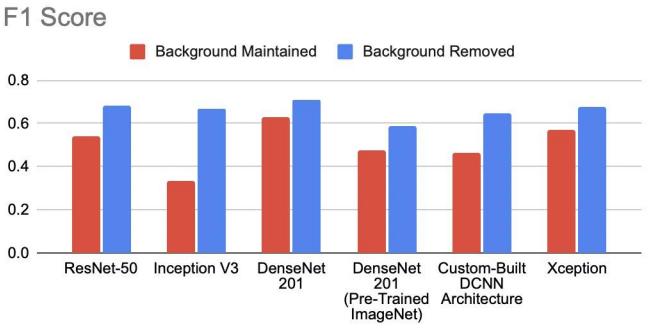

Figure VI.5: F1 Scores Comparison (With & Without Background Removed)

From the compiled data & charts above, it could be clearly determined that there is a significant change in every DCNN architecture's performances if the images backgrounds were maintained. On average, the Accuracy and F1 Scores depleted by 14% and 16% respectively. Even the best performing DenseNet 201 model, which previously attained an Accuracy of 76%, only managed to retain a mere 68.7%. Therefore, it could be concluded that neglecting irrelevant features from the dataset does help improve the performances of the architectures.

F. Impact of Removing Background - Accuracy & Loss Trends

The graphs below displays the aggregated average Accuracy & Loss values for each Train & Validation portions respectively:

Figure VI.6: Average Train & Validation Accuracy Scores Comparison (With & Without Background Removed)

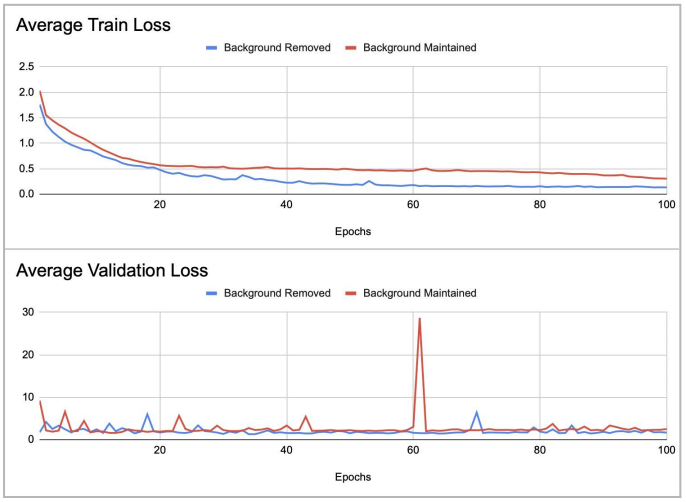

Figure VI.7: Average Train & Validation Loss Comparison (With & Without Background Removed)

The data plotted in the charts reveals that certainly removing image backgrounds do also improve the models' training performances on average. That is, they are able to achieve higher Accuracy Scores while having lower Loss values throughout the Epochs, for both the Train & Validation subsets.

G. Impact of Removing Background - Training Time Efficiency

The following visualization compares the Training Times per Epoch for each DCNN architecture, when the image backgrounds are maintained and removed:

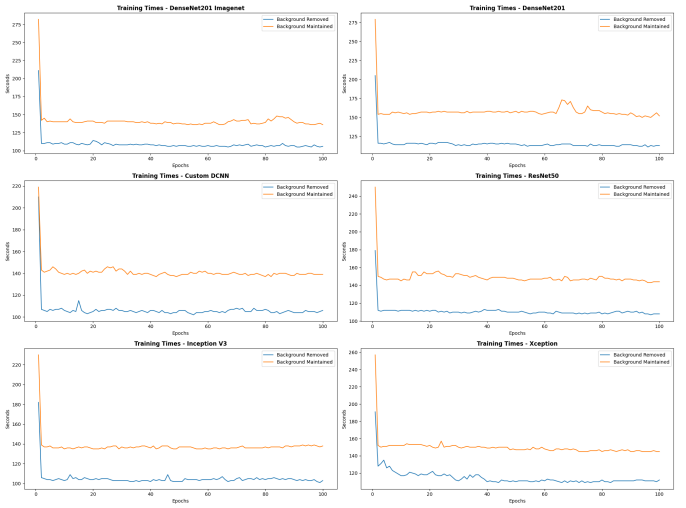

Figure VI.8: Training Time Trends per Epoch Comparison (With & Without Background Removed)

Overall, it could be clearly depicted that undoubtedly all the models are able to train much faster with the backgrounds removed. Looking closely, they are able to train less than 120 seconds without these unnecessary features, and would take up to over 150 seconds if otherwise.

Again, the Total Execution Times for each architecture with & without the image backgrounds removed will be computed and compiled below:

Table VI.9: Execution Times Comparison (With & Without Background Removed)

Model	Training Time (Hours) - Background Maintained	Training Time (Hours) - Background Removed	Change (%)
ResNet-50	4.27	2.96	-44.26
Inception V3	3.84	2.96	-29.73
DenseNet 201	4.45	3.26	-36.50
DenseNet 201 (Pre-Trained ImageNet)	3.98	3.21	-23.99
Custom-Built DCNN Architecture	3.94	2.73	-44.32
Xception	4.25	3.25	-30.77

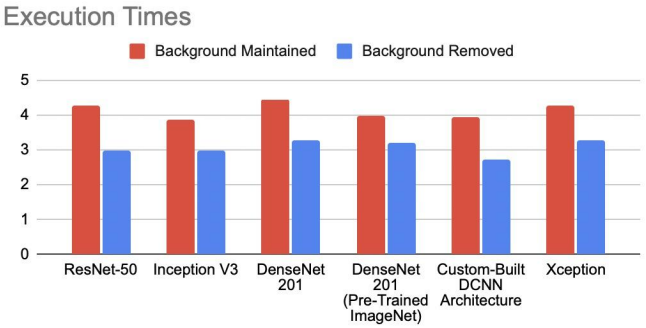

Figure VI.9: Execution Times Comparison (With & Without Background Removed)

Comparing the results summarized from table & figure above, there is clearly a significant reduction in the total Execution Times for every single model when its dataset image backgrounds were removed; decreasing by a range of 24 - 44% depending on the architecture. Therefore, just like the previous comparisons, this proves that removing the background does also have a positive impact on the models' overall Execution Times' efficiency..

VII. Conclusion

This research incorporates Computer Vision and Deep Learning for multiclass image classification on different automobile types. As revealed in the results and discussion, among the 6 different Deep Convolutional Neural Network architectures tested, the DenseNet 201 model is able to achieve the best performance in terms of Accuracy and F1 Scores of 76% and 0.71 respectively. Additionally, this research has also concluded that removing backgrounds from the dataset images, or in other words removing unnecessary noise from the input features, does have a significant positive impact on the

model's performances and Execution Times. On average, the Accuracy and F1 Scores surge by 14% and 16% respectively. Furthermore, the Execution Times have decreased by a range of 24 - 44% depending on the architecture. Hence, the results yielded from the conducted experiments has proven that Computer Vision had been successful in classifying different car types, and that Deep Learning could be helpful in automating the process of manually labeling cars, to assist alternative parking layout designs that improves the efficiency of space allocations in modern parking areas. Therefore, it could indirectly contribute to reducing parking congestion.

VIII. Recommendation

Although the solution of this experiment has been proven functional and thereby successful in assisting the research problem, its top performance metrics still lies in the seventies, which is considered decent but not exceptionally satisfactory. Hence, the models in this project could still be optimized further in many ways to achieve better performances; such as training it on larger & balanced car datasets, more or newer vehicle models (car manufacturers typically release new models every few years), tuning their hyperparameters, and many more. Likewise, this project is simply just a proof of concept for the extent of Deep Learning and Computer Vision in solving existing general issues; in the case of this research, assisting alternative parking layouts to reduce parking congestion.

IX. References

[1] McCoy, K., & TODAY, U. (2017, July 12). *Drivers spend an average of 17 hours a year searching for parking spots*. USA TODAY. https://www.usatoday.com/story/money/2017/07/12/parking-pain-causes-financial-and-personal-strain/467637001/

[2] *Real-time vehicle classification using CNN*. (2020, October 15). IEEE Xplore. https://ieeexplore.ieee.org/document/9225623

[3] *Convolutional neural network based vehicle classification in adverse Illuminous conditions for intelligent transportation systems*. (2021, February 13). Publishing Open Access research journals & papers | Hindawi. https://www.hindawi.com/journals/complexity/2021/6644861/

[4] Donny Avianto, Agus Harjoko, & Afia Hayati. (2022, October 22). *CNN-Based Classification for Highly Similar Vehicle Model Using Multi-Task Learning*. MDPI - Publisher of Open Access Journals. https://www.mdpi.com/2313-433X/8/11/293/pdf

[5] *Cars dataset*. Stanford Artificial Intelligence Laboratory. http://ai.stanford.edu/~jkrause/cars/car_dataset.html