ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE ENGINEERING AND TECHNOLOGY

CAR DAMAGE ANALYSIS FOR INSURANCE MARKET USING CONVOLUTIONAL NEURAL NETWORKS

M.Sc. THESIS

Cihat Tolga ARTAN

Department of Management Engineering

Management Engineering Program

JUNE 2019

ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE ENGINEERING AND TECHNOLOGY

CAR DAMAGE ANALYSIS FOR INSURANCE MARKET USING CONVOLUTIONAL NEURAL NETWORKS

M.Sc. THESIS

Cihat Tolga ARTAN
507961036

Department of Management Engineering

Management Engineering Program

Thesis Advisor: Assoc.Prof. Dr. Tolga KAYA

JUNE 2019

ISTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

SİGORTA SEKTÖRÜ İÇİN EVRİŞİMSEL SİNİR AĞLARI KULLANARAK ARAÇ HASARI ANALİZİ

YÜKSEK LİSANS TEZİ

Cihat Tolga ARTAN
(507961036)

İşletme Mühendisliği Anabilim Dalı

İşletme Mühendisliği Programı

Tez Danışmanı: Doç. Dr. Tolga KAYA

HAZİRAN 2019

Cihat Tolga ARTAN, a M.Sc. student of ITU Graduate School of Science Engineering and Technology student ID 507961036, successfully defended the thesis entitled "CAR DAMAGE ANALYSIS FOR INSURANCE MARKET USING CONVOLUTIONAL NEURAL NETWORKS", which he prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor :	**Doç. Dr. Tolga KAYA** İstanbul Technical University	.....
Jury Members :	**Doç. Dr. Tolga KAYA** İstanbul Technical University	.....
	Doç.Dr. Başar ÖZTAYŞİ Istanbul Technical University	.....
	Dr. Öğr. Üyesi Barış SOYBİLGEN Yeditepe University	.....
(If exists)	**Prof. Dr. Name SURNAME** ..... University	.....
(If exists)	**Prof. Dr. Name SURNAME** ..... University	.....

Date of Submission : 3 May 2019
Date of Defense : 14 June 2019

To my daughter Çağla,

FOREWORD

I would like to extend my thanks to my supervisor Assoc.Prof. Dr. Tolga KAYA for his valuable comments, advice and helpful guidance throughout the research.

My colleagues at LuckyEye, Deniz KARAKURT, Canan SARIÇİÇEK, Ece TANOVA, and Yunus Emre GAÇTI deserve many thanks for their contribution to my efforts during the research. Hereby, I would like to thank them for their valuable efforts.

Mehmet Abacı, CIO of Anadolu Sigorta, and the project team in Anadolu Sigorta shared valuable insights and enhanced my research for creating the business use case. I really would like to express my gratitude to them.

My family and my daughter Çağla have supported me during the whole study. Thank you…

June 2019

Cihat Tolga ARTAN
Industrial Engineer

TABLE OF CONTENTS

Page

ABBREVIATIONS

AI **:** Artificial Inteligence
CNN **:** Convolutional neural networks
CPU **:** Central Processing Unit
CV **:** Computer Vision
GPU **:** Graphical Processing Unit
ILSVRC **:** ImageNet Large Scale Visual Recognition Challenge
IoU **:** Intersection over Union
R-CNN : Region CNN
ReLU : Rectified Linear Unit
ResNet **:** Residual Network
R-FCN **:** Region Fully Convolutional Network
SSD **:** Single Shot Detector
SVM **:** Support Vector Machine

SYMBOLS

Lcls : Log Over two classes

R **:** Robost Loss Function

LIST OF TABLES

Page

LIST OF FIGURES

Page

CAR DAMAGE ANALYSIS FOR INSURANCE MARKET USING CONVOLUTIONAL NEURAL NETWORKS

SUMMARY

Artificial intelligence (AI) is a branch of computer science and technology aimed at developing the theories, methods, algorithms, and applications for assisting humans. AI helps people giving systems the problems directly and the systems learn how to solve them on their own. AI solutions are a rich set of algorithms used to perform AI tasks, applicable to perception and cognition that involve learning from data and experiences simulating human intelligence and capabilities. The ultimate goal of AI is to create technology that allows system to act in a highly intelligent manner.

Within artificial intelligence (AI), machine learning has come to the fore as the method of choice for developing practical software for computer vision, speech recognition, natural language processing, robot control, and other applications. The latest machine learning methods involve supervised learning systems like spam classifiers of e-mail, face recognizers over images, and medical diagnosis systems for patients; and they are a part of deep networks, which are multilayer networks of threshold units, each of which computes some simple parameterized function of its inputs.

Convolutional neural networks (CNNs) are deep artificial neural networks that are used primarily to classify images, cluster them by similarity, and perform object recognition within scenes. They are algorithms that can identify faces, individuals, street signs, tumors, platypuses and many other aspects of visual data.

With these capabilities of CNNs, different sectors use these models for different use cases to automate the processes which bring value by increasing the efficiency of the process. CNNs are enabling major advances in computer vision (CV), which has various applications for self-driving cars, robotics, drones, security, medical diagnoses, and treatments for the visually impaired

As the level of competition increases, image-based vehicle claim processing is gaining an important role in the insurance industry especially in handling small but more frequent insurance claims.

Considering these in this study, we used CNNs as a base for detecting the damaged areas of a car and then for classification of damage level of the detected area. We have used a two-step approach where we have used object detection and classification models together to understand the damaged area and the level of damage of a specific car using its images. The first step of our model is based on two different object detection models. Our selection criteria for these models is the accuracy of the model and the applicability in our usecase for the insurance industry. The second step of our study is based on classification models where we aim to classify the detected damage areas according to the level of damage (no damage, minor damage, and severe damage). In the classification model, we have also tried to extend our data set with new images for two different classes that are not performing well in terms of accuracy.

Our results show us that CNNs performs a good model for object detection and classification models. Based on the usecase, we can try different pre-trained models and train our own model with the help of these pre-trained models. In industries like insurance, it is very important to operate efficient systems to keep the costs as low as possible while maintaining good customer service.

SİGORTA SEKTÖRÜ İÇİN EVRİŞİMSEL SİNİR AĞLARI KULLANARAK ARAÇ HASARI ANALİZİ

ÖZET

Yapay zeka, bilgisayar biliminin bir alt dalı olarak insanlara yardım için; teorileri, yöntemleri, algoritmaları ve uygulamaları geliştirmeyi amaçlayan bir teknolojidir. Yapay zeka çözümleri, sistem problemlerini doğrudan algılayarak kendi başlarına problemleri nasıl çözeceklerini öğrenir. Yapay zeka çözümleri, verilen görevleri yerine getirmek için veriden öğrenmeyi bilen ve insan zekasını ve yeteneklerini simüle eden, deneyimlerden çıkarım yapmayı içeren zengin bir algoritma setidir. Yapay zekanın nihai amacı, bulunduğu sistemin çok akıllı bir şekilde hareket etmesini sağlayan bir teknoloji oluşturmaktır.

Yapay zeka içerisinde, makine öğrenimi; bilgisayarlı görüntü algılama, konuşma tanıma, doğal dil işleme, robot kontrolü ve diğer uygulamalar için pratik yazılımlar geliştirmek için tercih edilen yöntem olarak ortaya çıkmıştır. En yeni makine öğrenme yöntemleri, e-postaların spam sınıflandırıcıları, görüntülerin yüz tanıyıcıları ve hastalar için tıbbi tanı sistemlerinde kullanılmaktadır ve her biri girdilerinin basit parametrelerle işlevini hesaplayan çok katmanlı ağı olan derin ağların bir parçasıdır.

Evrişimsel Sinir Ağları (CNN'ler), öncelikle görüntüleri sınıflandırmak, benzerlikleri ile kümelemek ve görüntülerde nesne tanıma yapmak için kullanılan derin yapay sinir ağlarıdır. Yüzleri, bireyleri, sokak işaretlerini, tümörleri, örnekleri ve görsel verinin diğer birçok yönünü tanımlayabilen algoritmalardır.

Evrişimsel Sinir Ağlarının, özellikle 2010 yılında itibaren gerçekleştirilmekte olan uluslararası görsel tanıma yarışmaları ile hızla geliştiğini ve bu gelişime paralel olarak sadece obje tanıma alanında değil yapay sinir ağları ile ilgili birçok ihtiyaç alanında da kullanılmak üzere geliştirilmekte olduğunu inceledik. Bu yarışmalarda test edilen modeller ile ilgili bilimsel çalışmaları analiz ederek, hangi alanlarda gelişim gösterdiklerini ve yaşanan hangi problemlere çözüm sunmak üzere oluşturulduklarını öğrendik.

Bu çalışmada, CNN'leri bir otomobilin hasarlı alanlarını tespit etmek ve ardından tespit edilen alanın hasar seviyesini sınıflandırmak için bir yapay zeka modeli olarak kullandık. Zarar görmüş alanı ve görüntülerini kullanarak belirli bir otomobilin hasar seviyesini anlamak için birlikte nesne algılama ve sınıflandırma modellerini kullandığımız iki aşamalı bir yaklaşımı benimsedik. Modelimizin ilk adımı iki farklı nesne algılama modeline dayanıyor. Seçim kriterlerimiz modelin doğruluğu ve sigortacılık endüstrisinde uygulanabilirliği idi. Çalışmamızın ikinci basamağı, tespit edilen hasar alanlarını hasar seviyesine göre tespit etmeyi hedeflediğimiz sınıflandırma modellerine dayanmaktadır. Deneylerimiz sırasında sınıflandırma modelinde, doğruluk açısından iyi performans göstermeyen iki farklı sınıf için veri setimizi yeni görüntülerle genişletmeye çalıştık. Her iki deneyi, her iki veri modeli için

test edip, sonuçları karşılaştırarak sigortacılık sektörü için oluşturulabilecek bir çözümün oluşturulmasını sağladık.

Çalışmamızda CNN'lerin yapısına, gelişimlerine ve mimarilerini detaylı olarak inceleyerek, farklı kullanım alanları için kullanılabilecek CNN modelleri arasındaki farklılıkları anlamaya özen gösterdik. Bu sayede her eğitilmiş modelin hangi tür problemler için uygun olacağı konusunda hem teorik hem de yaptığımız deneyler aracılığı ile pratik bilgi sahibi olmayı amaçladık. Özellikle MobileNet gibi CNN modellerinin doğruluk açısından performanslarının benzer modeller ile karşılaştırılması fırsatı oluşturmayı hedefledik. MobileNet'in hızlı eğitilebilmesi ve daha az kaynak kullanmasının, diğer modellere göre doğruluk bazında ne ölçüde kayıplara yol açtığını gözlemledik. Bu sayede frklı arayüzler üzerinde gerçekleştirilebilecek yapay zeka modellerine yönelik performans için gözlemlerimiz oldu.

Obje tanıma modeli için iki farklı model kullandık. Faster R-CNN ve SSD. Bu modellerin seçiminde ve kullanımında daha önce gerçekleştirilmiş olan akademik makalelerdeki örnek veri kümeleri ile ilgili sonuçları inceledik. İncelemelerimiz çerçevesinde, kendi oluşturduğumuz veri setine uygun kurulum ayarlarını hazırlayarak obje tanıma deneylerimizi gerçekleştirdik. Bu iki model üzerinden geçirdiğimiz veri seti ile görsellerdeki araç bölgelerini tespit ettik. Çalışmamızda üç hasar bölgesi üzerinde modelleri eğittik ve deneylerimiz yaptık. Modelimiz, ilgili araç bölgesini belirli bir doğruluk payı ile tanımanın yanı sıra, tespit edilen bölgeyi resmin geri kalanından ayırarak; sınıflandırma modeli için hazırlık yapmamıza yardımcı oldu.

Sınıflandırma deneylerimiz için ise Inception v3 ve MobileNet eğitilmiş modelleri üzerinde çalıştık. Yaptığımız deneylerde farklı hasar bölgeleri için sonuçlar elde ettik ve bu sonuçları hata matrisi yaklaşımı ile analiz ettik. Yaptığımız analizlerde bazı hasar bölgeleri için Inception v3 bazlı modelin, bazı hasar seviyeleri için de MobileNET bazlı modelin hasar seviyesi tespiti için daha etkin çalıştığını gözlemledik. Özellikle düşük hasar seviyesi sınıfındaki sonuçların iyileştirilmesi amacı ile gerçekleştirdiğimiz deneylerde de benzer sonuçlar aldık.

Rekabet seviyesi arttıkça, görüntü tabanlı araç talep işleme, sigorta sektöründe özellikle küçük ama daha sık sigorta taleplerinin ele alınmasında önemli bir rol oynamaktadır.

Çalışmanın sonuçları çerçevesinde ele aldığımız dikey çözüm yaklaşımında verinin sayısının ve kalitesinin çok önemli olduğunu gözlemledik. Bu çalışma elde ettiğimiz sonuçların daha iyi veri kümeleri oluşturabilmek için imkân sağlayan yapay zeka destekli bir uygulamaya dönüştürülmesini öngörmekteyiz.

Sonuçlarımız bize CNN'lerin nesne algılama ve sınıflandırma çalışmaları için verimli modeller oluşturduğunu göstermektedir. İş ihtiyaçlarına dayanarak, önceden eğitilmiş farklı modelleri deneyebilir ve önceden hazırlanmış bu modeller yardımı ile kendi modelimizi eğitebiliriz. Sigorta gibi sektörlerde, iyi müşteri hizmetini sürdürürken maliyetleri mümkün olduğunca düşük tutmak için verimli sistemler kullanmak çok önemlidir. Benzer sektörlerde çalışan kuruluşların CNN'ler yardımıyla kendi modellerini yaratma yetenekleri gözlemlenmektedir.

Gerçekleştirdiğimiz çalışmadan sonraki aşamalarda farklı CNN modellerinin kullanılarak özellikle sigorta sektöründeki hasar maliyeti tahmin çalışmalarının farklı bakış açıları ile yeniden değerlendirilmesinin gerekli olduğunu düşünüyoruz. Hasar tespit sürecinin müşteri, sigorta şirketi ve hasarı onaran servis şirketi arasında geçen

sürecin detaylı analizi ile benzer yapay zeka uygulamalarının doğrudan veya ilişkili olarak farklı ara yüzlerde devreye alınmasının mümkün olacağını gözlemliyoruz. Özellikle veri açısından sıkıntı yaşamayan sigorta şirketlerinin veri kalitesini artırmak ve yapay zeka modellerinin iyi sonuçlar verebileceği düzeye getirmesinin sonraki çalışmalar açısından kritik olduğunu düşünüyoruz. Tüketicilerin yaşadığı problemlerin minimize edilerek, sürecin hızlandırılmasına yönelik çalışmalarda yapay zeka uygulamalarının ve de evrişimsel sinir ağlarının etkin biçimde kullanılabileceğini öngörüyoruz.

1. INTRODUCTION

Artificial intelligence (AI) is a branch of computer science and technology aimed at developing the theories, methods, algorithms, and applications for simulating and extending human intelligence. Today, AI assists people in giving system problems directly and the systems learn how to solve them on their own using a set of algorithms. AI algorithms are a rich set of algorithms used to perform AI tasks, applicable to perception and cognition that involve learning from data and experiences simulating human intelligence and capabilities. Among the most important capabilities is that of learning, which enables automated machine algorithms to improve through experiences by themselves.

Within AI, machine learning has progressed as the method for developing practical software for computer vision, speech recognition, natural language processing, robot control, and other applications. Many developers of AI systems now realize that, for many use-cases, it is easier to train a system by showing it examples of desired input-output behavior than to program it manually by anticipating the desired response for all possible inputs.

The most widely used machine-learning methods are supervised learning methods. Supervised learning systems, including spam classifiers of e-mail, face recognizers over images, and medical diagnosis systems for patients; involves deep networks, which are multilayer networks of threshold units, each of which computes some simple parameterized function of its inputs. With the advances in modern parallel computing architectures, such as graphics processing units originally developed for video gaming, it has been possible to build deep learning systems that contain billions of parameters and that can be trained on the very large collections of images, videos, and speech samples. Such large-scale deep learning systems have had a major effect in computer vision and speech recognition, where they have yielded major improvements in performance over previous approaches. Additionally, a large number of other real-world applications have been successfully put into practice due to deep learning, including image captioning, visual question answering, web search, natural language

processing, drug discovery and toxicology, customer relationship management, recommendation systems, medical informatics, Internet advertisements, medical image analysis, robotics, self-driving vehicles, board games.

Setting aside their huge empirical successes, models of neural-network-based deep learning are often simpler and easier to design and use the same model architecture in many different applications. Software toolkits have been developed to allow faster and more efficient implementation of these models. Today deep neural networks are the most used method of choice for a wide variety of machine-learning and AI tasks over large data sets.

Convolutional neural networks (CNNs) are deep artificial neural networks that are used primarily to classify images, cluster them by similarity, and perform object recognition within scenes. They are algorithms that can identify faces, individuals, street signs, tumors, platypuses and many other aspects of visual data.

CNNs perform optical character recognition (OCR) to digitize text and make natural-language processing possible on analog and hand-written documents, where the images are symbols to be transcribed. CNNs can also be applied to sound when it is represented visually as a spectrogram. More recently, convolutional networks have been applied directly to text analytics as well as graph data with graph convolutional networks.

The efficacy of CNNs in image recognition is one of the main reasons for the popularity of deep learning. They are powering major advances in computer vision (CV), which has obvious applications for self-driving cars, robotics, drones, security, medical diagnoses, and treatments for the visually impaired.

Recently, CNNs have surpassed humans on several tasks such as classification of natural objects and classification of traffic signs. After their great success, CNNs have become the first choice for learning features from training data.

One of the fields that were greatly influenced by CNNs is the automotive industry. Tasks such as pedestrian detection, car detection, traffic sign recognition, traffic light recognition, and road scene understanding are rarely performed by using hand-crafted features.

In the insurance industry, visual inspection and validation are currently used to reduce the efforts to understand the difference between the actual claim payment made and

the amount that should have been paid. However, these efforts cause delays in claim processing. Thus, fully or semi-automatic insurance claim processing could be very useful in the insurance industry when handling small but more frequent insurance claims that are under a certain amount of rate. It would increase the speed of claim investigation and loss assessment.

In this study, we used CNNs as a base for detecting the damaged areas of a car and then for classification of damage level of the detected area. Throughout our research, we classified 14 different damage areas for cars and selected three damage areas (bumper, fender, and hood) to evaluate in our study. Likewise during our research, we could not find out a publicly available dataset for car damage classification. Therefore, we created a dataset based on images taken from an insurance company and the internet. We manually tagged them according to the requirements of our study. The classification work took time and required great attention due to factors like the quality of the images, inter-class similarity, and visibility of damage level.

We used a two-step experiment approach. The aim of the first step was to detect the area of the car in a given picture so we used two object detection models, namely being Faster Region-CNN (Faster R-CNN) and Single Shot Detector (SSD), and we tried to evaluate the differences among them. In the second step, our aim was to classify the damage level for the damage area of the car that was selected via using the first step. For the second step, we used two different classification models to determine the level of selected damage (no damage, minor damage, and severe damage) for our transfer learning approach. We observed the results of two different models trained with Inception v3 and MobileNet. We observed that a combination of object detection and classification models for transfer learning in car damage analysis for the insurance market can be successful for different real life use cases.

The remainder of this paper is organized as follows. In Section 2, we present a review of the related work especially based on the CNNs we worked with. Later in Section 3, we describe our research approach. In Section 4, we explain the details of our experiments with selected models and share the results of our experiments in detail. We present our conclusions for the application of CNNs in the context of car damage analysis for the insurance market and possible future directions in Section 5.

2. LITERATURE REVIEW

Machine Learning is a multidisciplinary approach, involving several scientific domains (e.g. mathematics, computer science, physics, biology, etc.), that enable computers to automatically learn from data. Learning is a process that takes as input data and gives as output algorithms capable of performing, over the same kind of data, the desired task (Camastra & Vinciarelli, 2015). Machine Learning addresses the question of how to build computers that can automatically improve through experience. It is one of today's most rapidly growing technical fields, lying at the intersection of computer science and statistics, and at the core of artificial intelligence and data science (Jordan & Mitchell, 2015). Moreover, Machine Learning can be classified into several areas:

i unsupervised learning, in which a function is learned to describe patterns in the unlabeled training data,

ii reinforcement learning, in which successful strategies are learned from rewards and punishments (trial-and-error),

iii supervised learning, in which a computer program is learned from labeled training data in order to predict the true labels of 'new' data

The most widely used machine-learning methods are supervised learning methods. Supervised learning systems, including spam classifiers of e-mail, face recognizers over images, and medical diagnosis systems for patients, all exemplify the function approximation problem, where the training data take the form of a collection of (x, y) pairs and the goal is to produce a prediction y* in response to a query x*. The inputs x may be classical vectors or they may be more complex objects such as documents, images, DNA sequences, or graphs (Murphy, 2012).

Supervised learning systems generally form their predictions via a learned mapping f(x), which produces an output y for each input x (or a probability distribution over y given x). Many different forms of mapping f exist, including decision trees, decision forests, logistic regression, support vector machines, neural networks, kernel machines, and Bayesian classifiers (Hastie, Tibshirani, & Friedman, 2011).

One high-impact area of progress in supervised learning in recent years involves deep networks, which are multilayer networks of threshold units, each of which computes some simple parameterized function of its inputs (Schmidhuber, 2015). Deep learning systems make use of gradient-based optimization algorithms to adjust parameters throughout such a multilayered network based on errors at its output. Exploiting modern parallel computing architectures, such as graphics processing units originally developed for video gaming, it has been possible to build deep learning systems that contain billions of parameters and that can be trained on the very large collections of images, videos, and speech samples available on the Internet.

Different models for classification come from deep learning, which is one of the machine learning methods that are based on neural networks. Examples like speech recognition (Graves, Mohamed, & Hinton, 2013) using recurrent neural networks, natural language processing (Collobert, et al., 2011) using unified neural networks and visual recognition which will be explained in detail in our study, are typical AI tasks based on human performance.

Recognizing objects and localizing them in images is one of the most fundamental and challenging problems in computer vision. There has been significant progress on this problem over the decades due largely to the improvements in Deep Convolutional Networks (CNNs) (Krizhevsky, Sutskever, & Hinton, 2012).

The Ph.D. thesis of Jayawerdena, is one of the first studies in the literature that aims to automate vehicle damage detection using photographs by utilizing 3D computer-aided design (CAD) models of the photographed vehicle to identify how it would look before it was damaged (Jayawardena, 2013). As it is practically difficult to get 3D models of high quality, it is difficult to replicate his research. Since 2013 due to the great developments in the application of CNNs in computer vision, CNNs have performed well under computer vision tasks such as object recognition and detection.

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) acts as a benchmark for object recognition tasks (Russakovsky, et al., 2015). In 2012, SuperVision (now also known as AlexNet) has opened new doors for image recognition with their relatively small CNN model (Krizhevsky, Sutskever, & Hinton, 2012). Later in 2013 Clarifai decreased the stride and receptive field of the first convolutional layer and increased the number of filters of the middle convolutional layers (Zeiler, M. D. and

Fergus, R., 2014). VGG, the runner-up of 2014, was recognized the importance of CNN's depth for its performance with its simple but effective architecture and became an inspiration for many later architectures (Simonyan & Zisserman, 2014).

In 2014 GoogLeNet used the architecture of similar depth with more advanced inception modules and provided the base for later models (Szegedy, et al., 2014). In 2015 the residual network (ResNet) of Microsoft Research Asia (MSRA) used batch normalization intensely, following the practice of Ioffe and Szegedy that allowed deeper networks to converge (Ioffe & Szegedy, 2015). He et al. (2016) dealt with this problem using residual learning where their solution allowed MSRA to train a very deep CNN of over 150 layers (He, Zhang, Ren, & Sun, 2016)

The fast and accurate visual system of people allows us to perform complex tasks like driving with little conscious thought. Fast, accurate models for object detection would allow systems to drive cars without specialized sensors, enable auxiliary devices to transmit real-time scene information to humans, and unlock the potential for responsive robotic systems. Recent approaches like Region CNN (R-CNN) use region proposal methods to first generate potential bounding boxes in an image and then they run a classifier on the proposed boxes. After classification, post-processing is used to refine the bounding boxes, eliminate duplicate detections, and re-score the boxes based on other objects in the scene (Girshick, Donahue, Darrell, & Malik, 2014). The object detection system of R-CNN consists of three modules. The first generates category-independent region proposals. These proposals define the set of candidate detections available to the selected detector. The second module is a large convolutional neural network that extracts a fixed-length feature vector from each region. The third module is a set of class-specific linear SVMs.

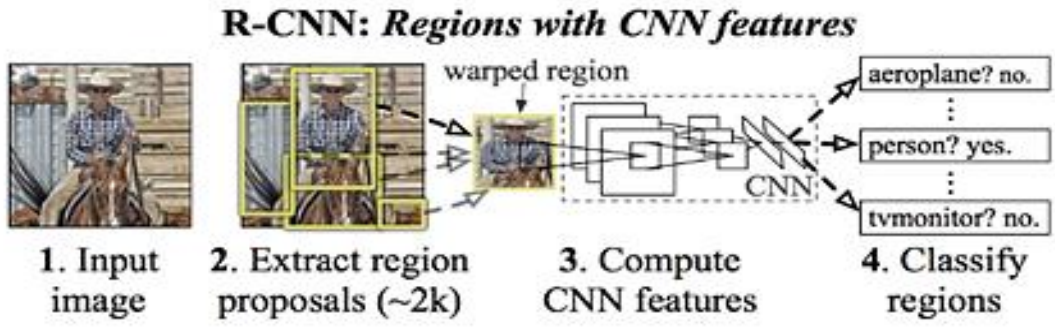

Figure 2.1 : Object detection system overview. (Girshick, Donahue, Darrell, & Malik, 2014)

R-CNN method takes the straightforward approach of cropping externally computed box proposals out of an input image and running a neural net classifier on these crops. However, this approach can be expensive because many crops are necessary, leading to significant duplicated computation from overlapping crops. Fast R-CNN handles this problem by pushing the entire image once through a feature extractor then cropping from an intermediate layer so that crops share the computation load of feature extraction (Girshick., 2015). Like R-CNN detector, Fast R-CNN‘s detector as shown below in Figure 2.2 also uses an algorithm like Edge Boxes to generate region proposals. Unlike R-CNN detector, which crops and resizes region proposals, Fast R-CNN detector processes the entire image. Whereas an R-CNN detector must classify each region, Fast R-CNN pools CNN features corresponding to each region proposal. Fast R-CNN is more efficient than R-CNN because in the Fast R-CNN detector, the computations for overlapping regions are shared.

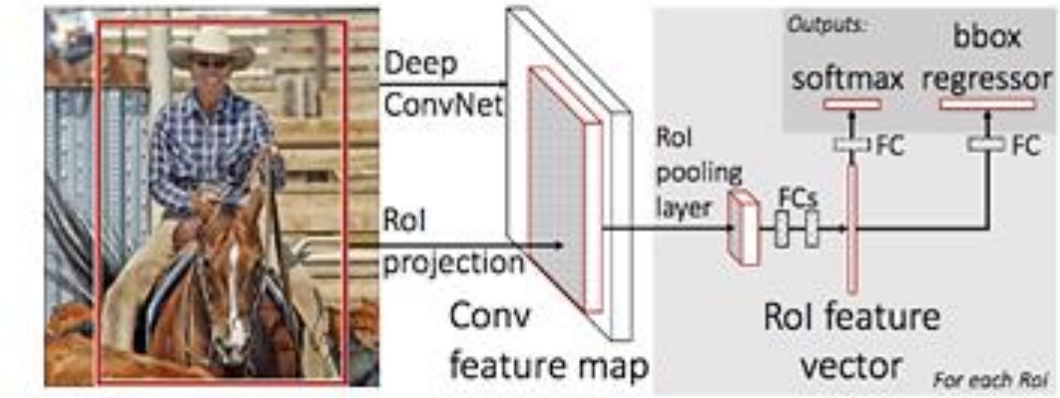

Figure 2.2 : Fast R-CNN combined with CNN. (Girshick, Donahue, Darrell, & Malik, 2014).

Towards the middle of 2015, a team at Microsoft Research found a way to make the region proposal step almost cost-free through an architecture they named Faster R-CNN (Ren, He, Girshick, & Sun, 2016). As shown in figure 2.3, Faster R-CNN is composed of two modules. The first module is a deep fully convolutional network that proposes regions (RPN), and the second module is the Fast R-CNN detector that uses the proposed regions.

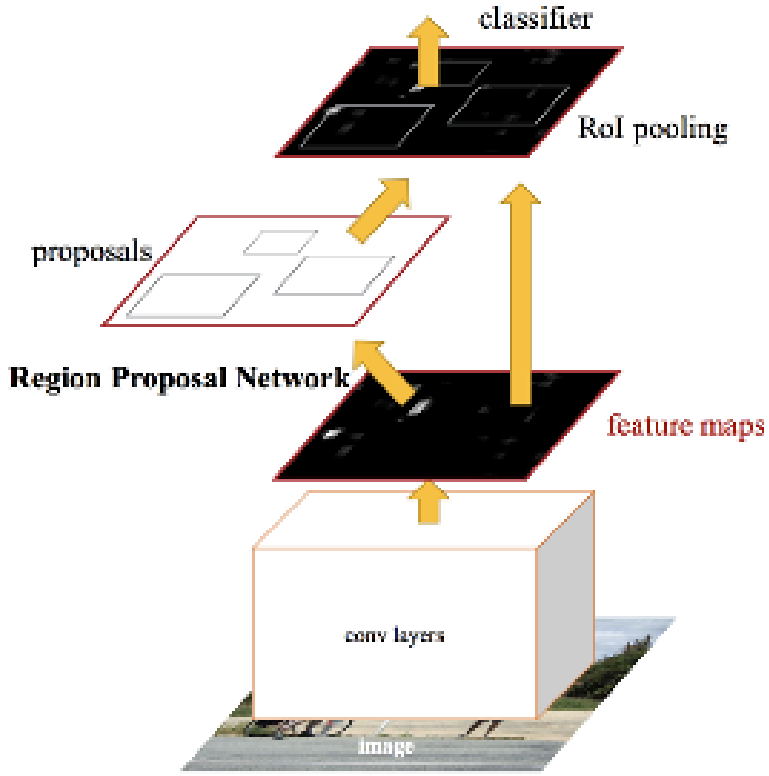

Figure 2.3 : Faster R-CNN and the region proposals. (Ren, He, Girshick, & Sun, 2016).

The entire system is a single, unified network for object detection (Figure 2.3). Using the recently popular terminology of neural networks with 'attention' mechanisms, the Region Proposal Network (RPN) module (Figure 2.4) tells the Fast R-CNN module where to look. Faster R-CNN adds a Fully Convolutional Network on top of the features of the CNN creating what is known as the Region Proposal Network.

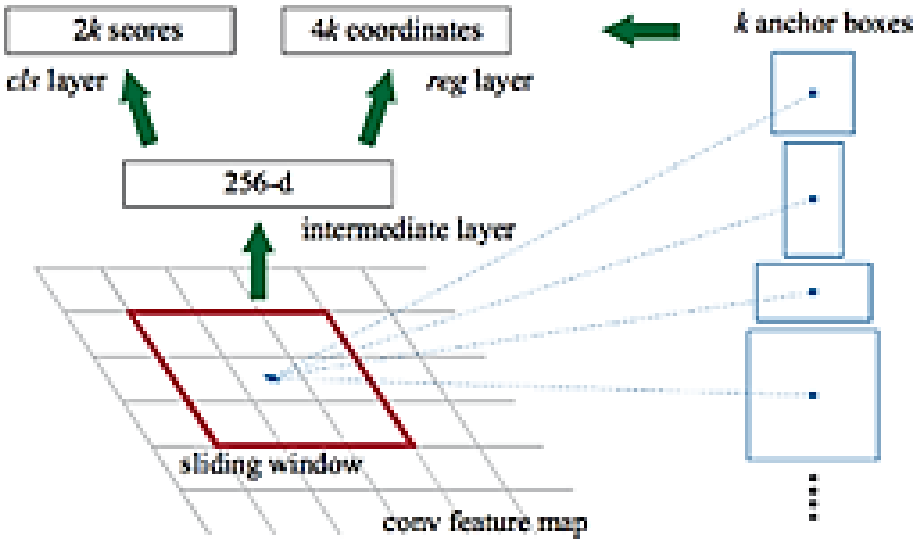

Figure 2.4 : The Region Proposal Network (Ren, He, Girshick, & Sun, 2016).

In Region-based Fully Convolutional Network (R-FCN) which uses positive sensitive score map, the inference time is much faster than Faster R-CNN while still maintaining competitive accuracy (Dai, Li, He, & Sun, 2016).

In 2016, Single Shot Detector (SSD) was presented for detecting objects in images using a single deep neural network by decoupling the output space of bounding boxes into a set of default boxes over different aspect ratios and scales per feature map location (Liu, 2016). While predicting, the network generates scores for the presence of each object category in each default box and produces adjustments to the box to better match the object shape.

Selecting a detection architecture that achieves the right speed/memory/accuracy balance for a given application and platform is very important in real life use cases for modern convolutional object detection systems. A recent study by J.Huang et al. (2017) provides a concise survey of modern convolutional detection systems, and describes how the leading ones follow similar designs by defining flexible and unified implementation of the three meta-architectures (Faster R-CNN, R-FCN, and SSD) in Tensorflow which they use to do extensive experiments that trace the accuracy/speed tradeoff curve for different detection systems, varying meta-architecture, feature extractor, image resolution (Huang, et al., 2017). Their findings show that using fewer proposals for Faster R-CNN can speed it up significantly without a big loss in accuracy, making it competitive with its faster cousins, SSD and R-FCN while SSDs performance is less sensitive to the quality of the feature extractor than Faster R-CNN and R-FCN.

3. METHODOLOGY

In our methodology, we used an iterative process with multiple phases.

• **Understanding The Business Requirements:** We worked with one of the top 3 insurance companies in Turkey to analyze how we can create value with our research for their claim analysis process. We met several times face to face with Subject Matter Experts (SMEs) for damage analysis and cost evaluation and we had different use cases for their internal and external processes.

• **Data Understanding:** We studied with SMEs to understand how they evaluate the damage process from pictures and how we can engage with their systems to create our datasets.

• **Data Preparation:** The data preparation phase was the most time consuming one as we combined the data that we gathered from SMEs of the insurance company with the data from search engines in order to enhance the quality of our datasets.

• **Modeling:** After we prepared our datasets, we split our data into separate sets for training, validation, and testing. The training data was used to determine our learning approach, validation data to optimize the parameters on the training data and use testing data to assess the resulting model.

• **Evaluation:** This process was conducted during the modeling phase to develop a model with high quality from a data analysis perspective.

• **Deployment:** We created a strategy to integrate our efforts for the use cases that we studied during the business requirement assessments.

With our research, we wanted to use our results in developing a real-life use case which can be tested in the insurance company we have collaborated.

We used an iterative process with multiple tasks that we have visualized in figure 3.1:

Task 1: Recognizing the damage areas of the car.

We found out that there are possible 14 damage areas for a car. We selected 3 damage areas (bumper, hood, fenders) for which we had a chance to discuss the damage

analysis details with the SMEs in the insurance company. Also, we could only get the images of the damaged cars based on these 3 damage areas from the SMEs.

Task 2: Classifying the level of damage in the selected area of the damaged car.

After getting the results of Task 1 with our model, we decided to use 3 classes for the evaluation of the level of damage:

No Damage (ND): No damage in the selected damage area,

Minor Damage (MD): Minor damage typically means scratches, scrapes or dings. For example, a cracked headlight or small dent in your hood.

Severe Damage (SD): Severe damage is just like it sounds. Very heavy damage. This includes broken axles and bent or twisted frames. In a severe damage situation, air bags have almost always deployed. Examples include if your car has rolled over, or a complete side of the car is damaged.

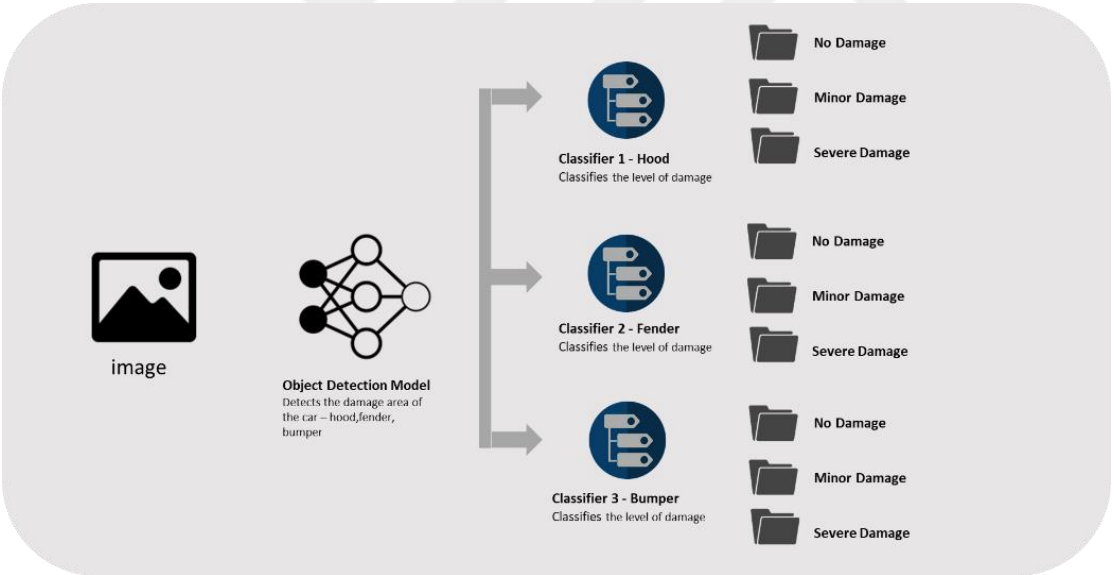

Figure 3.1 : Damage Analysis model framework

You can see the examples of our dataset in Figure 3.2 where we have separate images for the class for each damage level in each damage area.

Figure 3.2 : Examples from each class of damage area from our dataset.

In our study, one of the areas we focus on CNNs is classification. A classifier is a program that implements a so-called score function. In a given data instance, it computes a score for all C possible *classes*. The class with the highest score is predicted to be the true class for that data instance. An algorithm that returns a classifier based on a set of labeled training data is called a learner. To obtain some structure in choosing a learner from the countless amounts of possibilities, Domingos (2012) characterizes learning as a combination of three components: Representation, evaluation, and optimization.

A classifier must be represented in some formal language that the computer can handle. The representation of the learner is its most characteristic component and choosing a representation for a learner is identical to choosing the set of classifiers that it can possibly learn. A CNN is represented by a set of neurons ordered in one or more layers that are connected in a feed-forward manner without any cycles. Figure 3.3. shows an example of a regular neural network with a depth of 3 (the number of non-input layers), where 2 hidden layers are of 4 neurons.

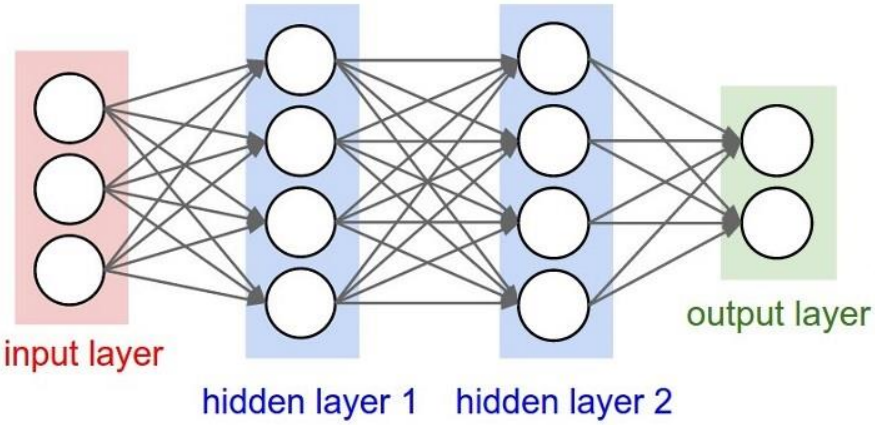

Figure 3.3 : A multi-layer NN with three inputs

Multi-layer neural networks only make sense when the activation functions used are non-linear. Cybenko (1989) shows that we can arbitrarily well approximate any function using a neural network with only one hidden layer and any continuous sigmoidal ('S-shaped') activation function. Ramachandran, Zoph, & Le (2017) find in a recent study that the current most successful and widely-used activation function is the rectified linear unit (ReLU) function f(x) = max(0,x) introduced by Nair & Hinton (2010). In the same study, Ramachandran, Zoph, & Le (2017) also describes Swish function $f(x) = x \cdot \sigma(x)$, which is similar to the ReLU function (Figure3.4), which has advantages of being smooth and have a non-zero left tail derivate.

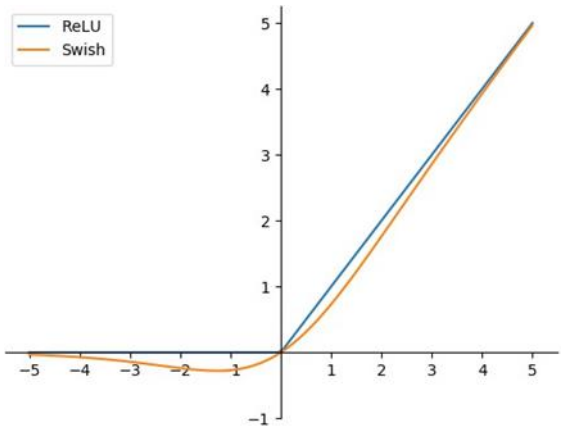

Figure 3.4 : The ReLU and Swish function

We want to explain different type of layers existing in CNNs. Input layer contains one neuron for every feature we use in our data set. In visual recognition, using the raw Red-Green-Blue pixel values is highly common. For regular neural networks, the input layer is usually followed by one or more fully-connected layers. The last fully-connected layer is called the output layer and it represents class scores for correct class

prediction. As the number of weights quickly explodes when using image data, other methods are used to reduce the number of weights in the first layer. In the convolutional layer, explained by (LeCun, et al., 1989), the number of weights is reduced by linking each node to only a limited part of its layer's input. Every node in this layer corresponds to a filter (or 'kernel') that connects it to a local region of the image, as shown in Figure 3.5. Every filter is used multiple times by sliding it over the whole input volume specified by the stride S, determining the number of pixels we move at a time, and the amount of zero-padding P on the border which can be used to control the width and height of the layer's output volume.

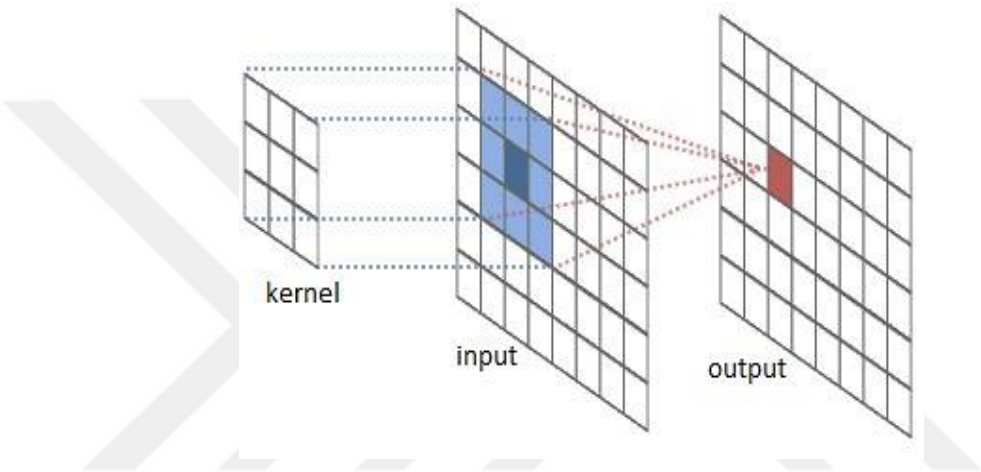

Figure 3.5 : A Convolutional filter

Similar to the convolutional layer, the pooling layer connects nodes to local regions of the input, using some receptive field *F* and stride *S* (usually no zero-padding). Instead of computing one or more filters, a pooling layer just computes a fixed function of each connected region, independently per depth slice and without taking any dot products first.

An evaluation function (also called objective function or scoring function) is needed to separate good classifiers from bad ones. The evaluation function used internally by the algorithm may differ from the external one that we want the classifier to optimize, for ease of optimization (Domingos, 2012). We don't need to optimize the training performance of the model in full as sometimes this can cause overfitting that can be described as the act of fitting the noise of individual training examples instead of the underlying patterns hidden in the full dataset. We can solve this problem in splitting the dataset into three separate sets (Figure 3.6) where a training set to optimize the parameters of a specified model; a validation set to provide an unbiased evaluation of

the resulting classifier and a test set to obtain an unbiased evaluation of the specified classifier in full. (Ripley, 2007)

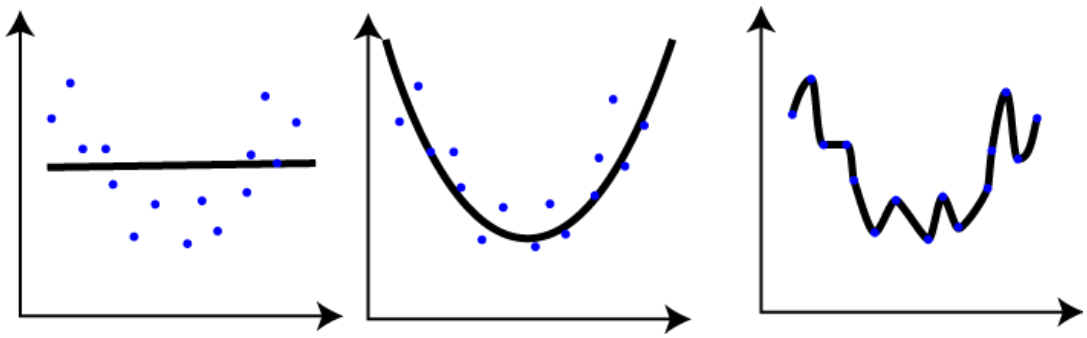

Figure 3.6 : The figure on the left shows under-fitting where a line is not flexible enough to capture the underlying pattern of dataset; the figure on the middle shows a parabola that follows the general pattern well with low complexity and the figure on the right shows a polynomial following the data perfectly with more complexity that can result in poor results in predicting new data.

Instead of reserving a part of the training data for validation, splitting the data into k sets and use each of them while training the other data sets is called cross-validation. With cross-validation, we do not lose part of the training set, yet more computational power is required for training.

We need to decide what evaluation function to use for comparing the performance of different classifiers. The evaluation function used by the learner internally for optimization is called Loss Function and it should be similar to the evaluation function used externally for reporting the performance of the resulting classifier. Whereas the loss function needs to be suitable for optimization, the external evaluation function should primarily give an easy way to interpret the measure of the model's performance. Batista, Prati, & Monard, (2004) observe that external evaluation functions are mostly based on confusion matrix elements where the most well-known is accuracy, measuring the fraction of all date instances predicted correctly.

		Predicted Class: Positive	Predicted Class: Negative
Actual Class	Positive	True Positive (# of TPs)	False Negatives (# of FNs)
	Negative	False Positives (# of FPs)	True Negatives (# of TNs)

Figure 3.7 : Confusion matrix summarizes the performance of a classifier for two classes

In some cases, accuracy (3.1) gives a biased representation of the performance, especially when the dataset is unbalanced or the majority of the instances is of one class (Figure 3.7). In that case, F_1-score (3.4), the harmonic mean of recall (3.2) and precision (3.3), the fraction of true predictions actually being true can be used. (Chawla, 2009)

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.1}$$

$$Recall = \frac{TP}{TP + FN} \tag{3.2}$$

$$Precision = \frac{TP}{TP + FP} \tag{3.3}$$

$$\mathrm{F1} = 2.\frac{Recall.Precision}{Recall + Precision} \tag{3.4}$$

3.1.Recognizing The Damage Area Of The Car

During our business analysis phase, we understood that in classifying the level of damage, understanding in which area the damage occurred is an important factor for the success of our model which lead us to study the object detection models in detail. For this end, we analyze the recent object detection models that can contribute to our experiment in terms of speed/memory/accuracy balance. Similarly, Huang, et al., (2017), studied selecting a detection architecture that achieves the right

speed/memory/accuracy balance for a given application and platform. They presented a unified implementation of the Faster R-CNN, R-FCN and SSD systems, which they viewed as "meta-architectures" and trace out the speed/accuracy trade-off curve created by using alternative feature extractors and varying other critical parameters such as image size within each of these meta-architectures.

Their findings show that using fewer proposals for Faster R-CNN can speed it up significantly without a big loss in accuracy, making it competitive with its faster cousins, SSD and R-FCN. They show that SSD's performance is less sensitive to the quality of the feature extractor than Faster R-CNN and R-FCN. They see that SSD models with Inception v2 and Mobilenet feature extractors are most accurate of the fastest models. Faster R-CNN with dense output Inception Resnet models attains the best possible accuracy on their optimality frontier, achieving the state-of-the-art single model performance. Based on this study and our use case for the insurance market, we decide to use SSD with MobileNet feature extractor as the first "meta-architecture" to train our model and Faster R-CNN with Inception 2 feature extractor for the second "meta-architecture".

3.1.1. Faster R-CNN

Faster R-CNN is composed of 2 modules. The first is a deep fully convolutional network that proposes regions and the second module is the Fast R-CNN detector that uses the proposed regions. The entire system is a single, unified network for object detection. The RPN module tells the Fast R-CNN module where to look. We use the standard loss function for an image that is defined as :

$$(\{p_i\}\{t_i\}) = \frac{1}{N_{\text{cls}}} \sum_i L_{\text{cls}} (p_i - p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i - t_i^*) \qquad (3.5)$$

Where i is the index of an anchor in a mini-batch and pi is the predicted probability of anchor i being an object. The ground-truth label p∗i is 1 if the anchor is positive, and is 0 if the anchor is negative. ti is a vector representing the 4 parameterized coordinates of the predicted bounding box, and t∗i is that of the ground-truth box associated with a positive anchor. The classification loss Lcls is log loss over two classes (object vs. not object). For the regression loss, they use Lreg(ti,t∗i) = R(ti − t∗i) where R is the robust loss function (smooth L1) defined in [2]. The term p∗iLreg means the regression

loss is activated only for positive anchors (p∗i = 1) and is disabled otherwise (p∗i = 0). The outputs of the cls and reg layers consist of {pi} and {ti} respectively. The two terms are normalized by N_{cls} and N_{reg} and weighted by a balancing parameter λ.

3.1.2. SSD: Single Shot MultiBox Detector

We selected SSD as the second meta-architecture for our model. SSD, discretizes the output space of bounding boxes into a set of default boxes over different aspect ratios and scales per feature map location. At prediction time, the network generates scores for the presence of each object category in each default box and produces adjustments to the box to better match the object shape. Additionally, the network combines predictions from multiple feature maps with different resolutions to naturally handle objects of various sizes. SSD is simply relative to methods that require object proposals because it completely eliminates proposal generation and subsequent pixel or feature resampling stages and encapsulates all computation in a single network. This makes SSD easy to train and straightforward to integrate into systems that require a detection component.

We have used the standard loss function for SSD which is a weighted sum of the localization loss (loc) and the confidence loss (conf):

$$L(x, c, l, g) = \frac{1}{N} L_{conf}(x, c) + \alpha L_{loc}(x, l, g) \tag{3.6}$$

where N is the number of matched default boxes. If $N = 0$, wet set the loss to 0. The localization loss is a Smooth L1 loss [6] between the predicted box (l) and the ground truth box (g) parameters. Similar to Faster R-CNN, we regress to offsets for the center (cx,cy) of the default bounding box (d) and for its width (w) and height (h)

$$L_{loc}(x, l, g) = \sum_{i \in Pos}^{N} \sum_{m \in \{cx,cy,w,h\}} x_{\mathrm{ij}}^{\mathrm{k}} smooth_{\mathrm{LI}} \left(x_i^m - g_j^m\right) \tag{3.7}$$

$$g'^{cx}_j = \left(g_j^{cx} - d_i^{cx}\right)/d_i^w \tag{3.8}$$

$$g'^{cy}_j = \left(g_j^{cy} - d_i^{cy}\right)/d_i^h \tag{3.9}$$

$$g'^{w}_{j} = log\left(\frac{g^{w}_{j}}{d^{w}_{i}}\right) \tag{3.10}$$

$$g'^{h}_{j} = log\left(\frac{g^{h}_{j}}{d^{h}_{i}}\right) \tag{3.11}$$

The confidence loss is the softmax loss over multiple classes confidences (*c*).

$$L_{con\,f}(x, c) = -\sum_{i \in Pos}^{N} x^{p}_{ij} \log\left(c'^{p}_{i}\right) - \sum_{i \in Neg} \log\left(c'^{0}_{i}\right) \tag{3.12}$$

$$\text{Where } c'^{p}_{i} = \frac{\exp(c^{p}_{i})}{\sum_{p} \exp(c^{p}_{i})} \tag{3.13}$$

and the weight term α is set to 1 by cross-validation

3.2. Classifying The Level of Damage

Image classification is the task of classifying a given image into one of the pre-defined categories. Traditional pipeline for image classification involves two modules: viz. feature extraction and classification. Feature extraction involves extracting a higher level of information from raw pixel values that can capture the distinction among the categories involved.

For the classifying part of our model, we have evaluated two pre-trained models Inception V3 and MobileNet.

3.2.1. Inception V3

The "Inception" micro-architecture was first introduced by (Szegedy, et al., 2014). The goal of the inception module is to act as a "multi-level feature extractor" by computing 1×1, 3×3, and 5×5 convolutions within the same module of the network. The output of these filters is then stacked along the channel dimension and before being fed into the next layer in the network.

The Inception V3 architecture included in the Keras core comes from the later publication by Szegedy et al., (Szegedy, Vanhoucke, Ioffe, Shlens, & Wojna, 2015)

which proposes updates to the inception module to further boost ImageNet classification accuracy. The weights for Inception V3 are smaller than both VGG and ResNet, coming in at 96MB. Based on the results of Inception v3 in this study, we decided to use Inception V3 with trained on Coco Dataset as the base of our classifier model.

3.2.2. MobileNet

The general trend has been to make deeper and more complicated networks in order to achieve higher accuracy. However, these advances to improve accuracy are not necessarily making networks more efficient with respect to size and speed. In many real-world applications such as robotics, self-driving car and augmented reality, the recognition tasks need to be carried out in a timely fashion on a computationally limited platform (Howard, et al., 2017). MobileNet model is an efficient network architecture and a set of two hyper-parameters in order to build very small, low latency models that can be easily matched to the design requirements for mobile and embedded vision applications.

We decided to use MobileNet in our damage level classifier based on the study by A.G.Howard et al., 2017 as the results of the experiment are very similar to Inception V3 nearly 10 times less compute-intensive that can help us to realize the mobility requirements of the insurance market for the Stanford Dogs test set (Howard, et al., 2017)

4. APPLICATION AND RESULTS

4.1.Experimental Setup

We perform the actual experiments according to our research approach starting with the recognition of the damage area of the car and then classifying the level of damage. Additionally, we experimented two different pre-trained models for each task on GPU based servers (4 servers with NVidia Tesla P 100 x 4 x 20 Core, 32 core automatic selected CPU) in Google and IBM Cloud Platforms. All the experiments were conducted by developing applications using Phyton in Tensorflow and dockerized for Cloud Computing.

4.2.Data

We used 2 different data sources to create our datasets.

a. The images we received from the insurance company,

b. The images we obtain by crawling Google Image Search.

We obtained two datasets:

1. Images with undamaged cars mostly collected through Google Search.

2. Images with damaged cars where most of the images are taken from the insurance company. These images include damages in 3 specific areas of the car. (bumper, fender, and hood) We manually tagged them according to the requirements of our experiments. The tagging and boxing work took time and great attention due to factors such as the quality of the images, inter-class similarity, and visibility of damage level.

Our final dataset includes 1.096 images and we have assigned 50 % for training, 30 % for validation and 20 % test.

Table 4.1 : Assignment of 1.096 car images

	Train (50%)	Validation (30%)	Test (20%)
Selected Images	549	328	219

In order to detect the damaged area and classify the level of the damage as well, we manually tagged the resulting dataset, which is summarized in Table 4.2. Then we manually defined the area of the car with bounding boxes.

Table 4.2 : Number of tags for each damage area of the car

Damage Area of The Car	Train (50%)	Validation (30%)	Test (20%)	Total
Bumper	462	285	223	967
Hood	473	281	174	928
Fenders	941	589	341	1.871
TOTAL	1.876	1.155	738	

4.3.Results for Recognizing The Damage Area Of The Car

Since 2010, the method of computing AP by the PASCAL VOC challenge has changed. Currently, the interpolation performed by PASCAL VOC challenge uses all data points, rather than interpolating only 11 equally spaced points as stated in their paper (Everingham, Van Gool, Wiliams, Winn, & Zisserman, 2010). As we want to reproduce their default implementation, our default code (as seen further) follows their most recent application (interpolating all data points).

Intersection over Union (IoU) is simply an evaluation metric. Any algorithm that provides predicted bounding boxes as output can be evaluated using IoU. In order to apply Intersection over Union to evaluate an (arbitrary) object detector we need:

i. The ground-truth bounding boxes (i.e., the hand-labeled bounding boxes from the testing set that specify where in the image our object is).

ii. The predicted bounding boxes from our model.

$$IoU = \frac{Area\ of\ Overlap}{Area\ of\ Union} \tag{4.1}$$

The mean Average Precision (mAP) is used for evaluating detection algorithms. The mAP metric is the product of precision and recall of the detected bounding boxes. The mAP value ranges from 0 to 100. The higher the number, the better it is. The mAP can

be computed by calculating average precision (AP) separately for each class, then the average over the class.

4.3.1. Results for Faster R-CNN

In our experiments with Faster R-CNN, for IoU is selected as 0.5, mAP for all three damage areas is 73%. Our classifier performs better for Hood than the other two classes. If we select IoU as 0.75, the overall mAP for our model decreases to 52%.

Table 4.3 : Average Precision for Faster R-CNN.

	Bumper AP	Hood AP	Fender AP	mAP
IoU = 0.5	68%	77%	73%	73%
IoU = 0.75	46%	58%	51%	52%

As you can see in Figure 4.1, our model is performing with 0.9 precision until we have recall value of 0.6 and then decreases rapidly as the recall rate increases for Bumper class. For the hood class, our model is performing with 0.9 precision until we have recall value of 0 .75 and then decreases rapidly as the recall rate increases. For the fender class, our model is performing with more than 0.9 precision until we have recall value of 0.75.

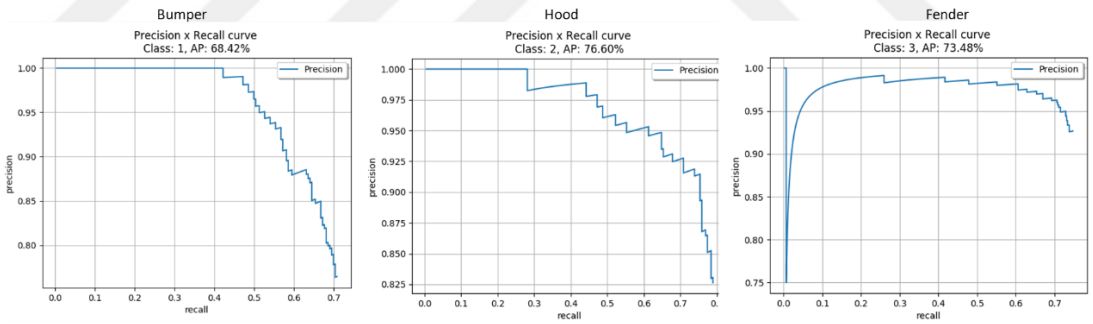

Figure 4.1 : Results for Faster R-CNN for IoU =0.5.

As you can see in Figure 4.2, when we increase IoU to 0.75; our model is performing with 0.65 precision until we have recall value of 0.5 for the bumper class. For the hood class, our model is performing with 0.8 precision until we have recall value of 0 .65 and then decreases rapidly as the recall rate increases. For the fender class our model is performing with more than 0.8 precision until we have recall value of 0.55 .

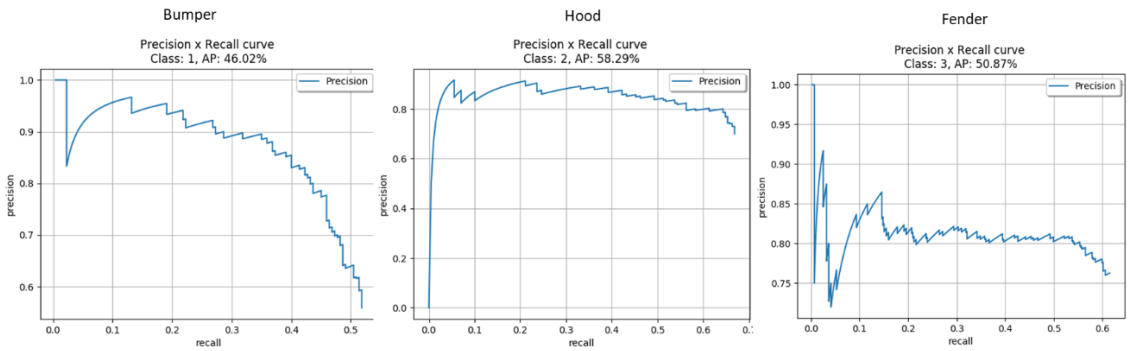

Figure 4.2 : Results for Faster R-CNN for IoU=0.75.

In Figure 4.3, you can see some examples from the results of our model for classification. As you can see from the bottom left images, our model needs better training for the rear bumper which is related to the number of images for rear bumper.

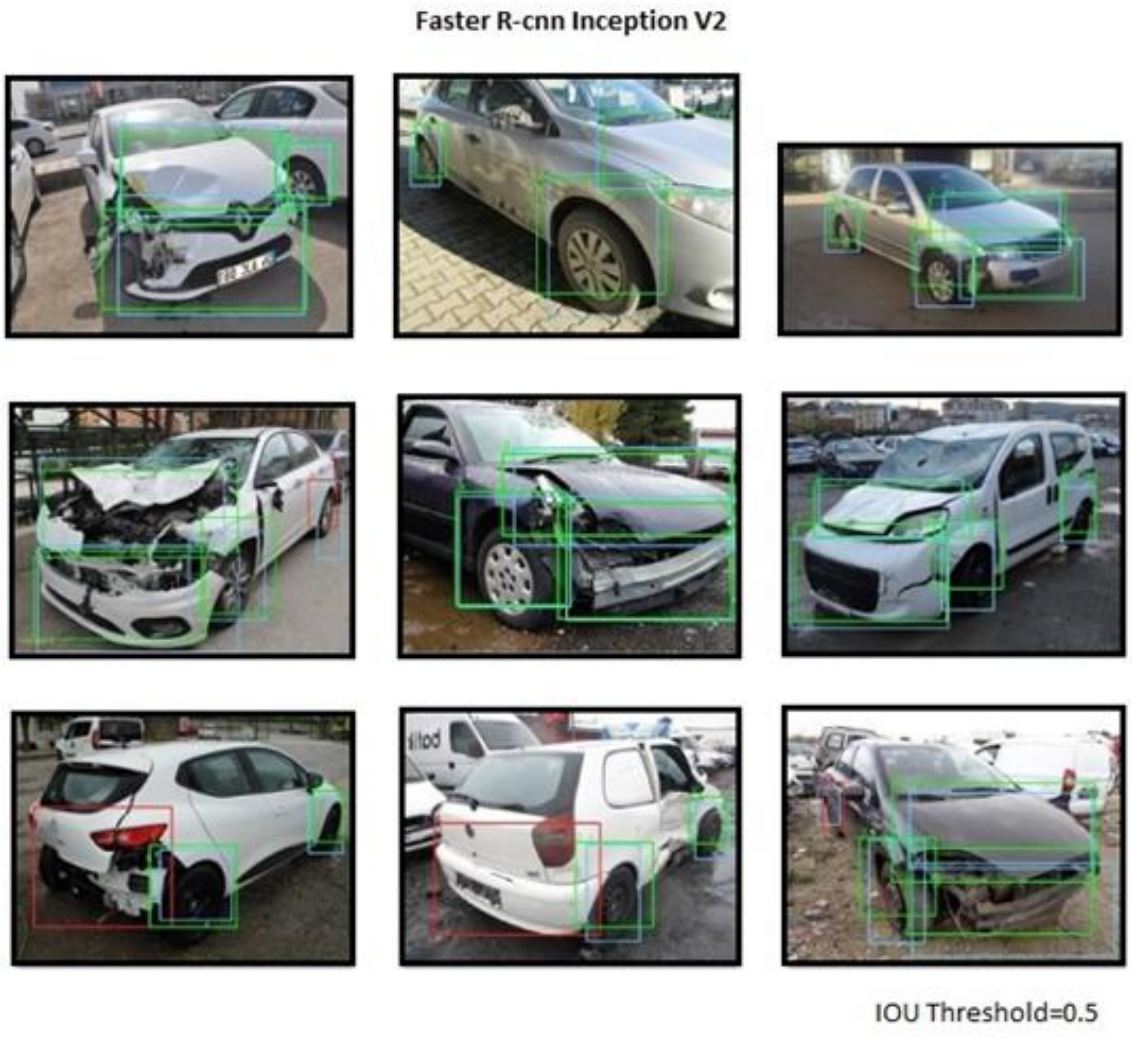

Figure 4.3 : Examples from the results of the experiment with Faster R-CNN.

4.3.2. Results for SSD

In our experiments with SSD, for IoU is selected as 0.5, mAP for all three damage areas is 31.76%. Our classifier performs better for hood than the other two classes. If we select IoU as 0.75, the overall mAP for our model decreases to 21.86%.

Table 4.4 : Average Precision for SSD.

	Bumper AP	Hood AP	Fender AP	mAP
IoU = 0.5	28.61%	41.69%	24.99%	31.76%
IoU = 0.75	20.15%	26.12%	19.30%	21.86%

In Figure 4.4, our model is performing with 0.7 precision until we have recall value of 0.6 for the bumper class. For the hood class our model is performing with 0.7 precision until we have recall value of 0.6. For the fender class our model is performing with more than 0.65 precision until we have recall value of 0.35.

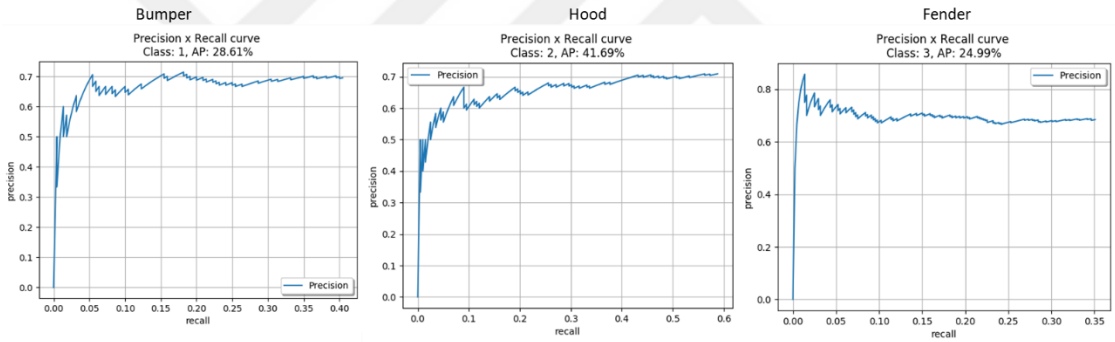

Figure 4.4 : Results for SSD for IoU =0.5 .

In Figure 4.5, when we increase IoU to 0.75; our model is performing with 0.6 precision until we have recall value of 0.3 for the bumper class. For the hood class, our model is performing with 0.5 precision until we have recall value of 0 .45 . For the fender class, our model is performing with more than 0.6 precision until we have recall value of 0.30 .

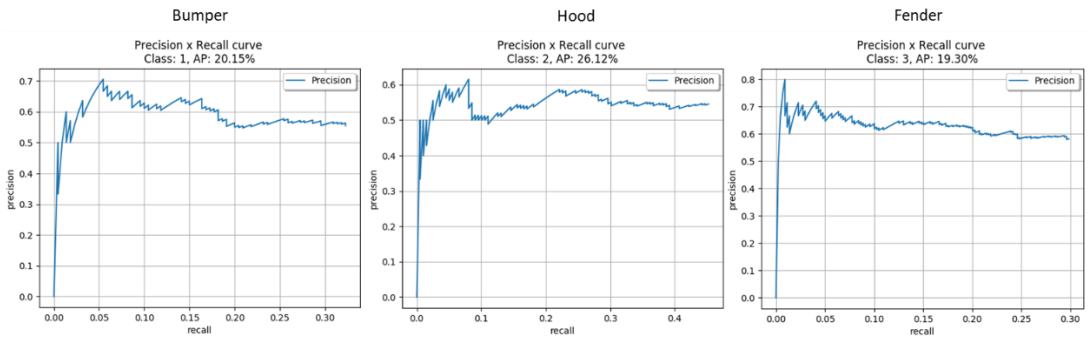

Figure 4.5 : Results for SSD for IoU =0.75 .

In Figure 4.6, you can see some examples from the results of our model with SSD for classification. Our model needs better training for all the damage areas which is related to the number of images for each of them.

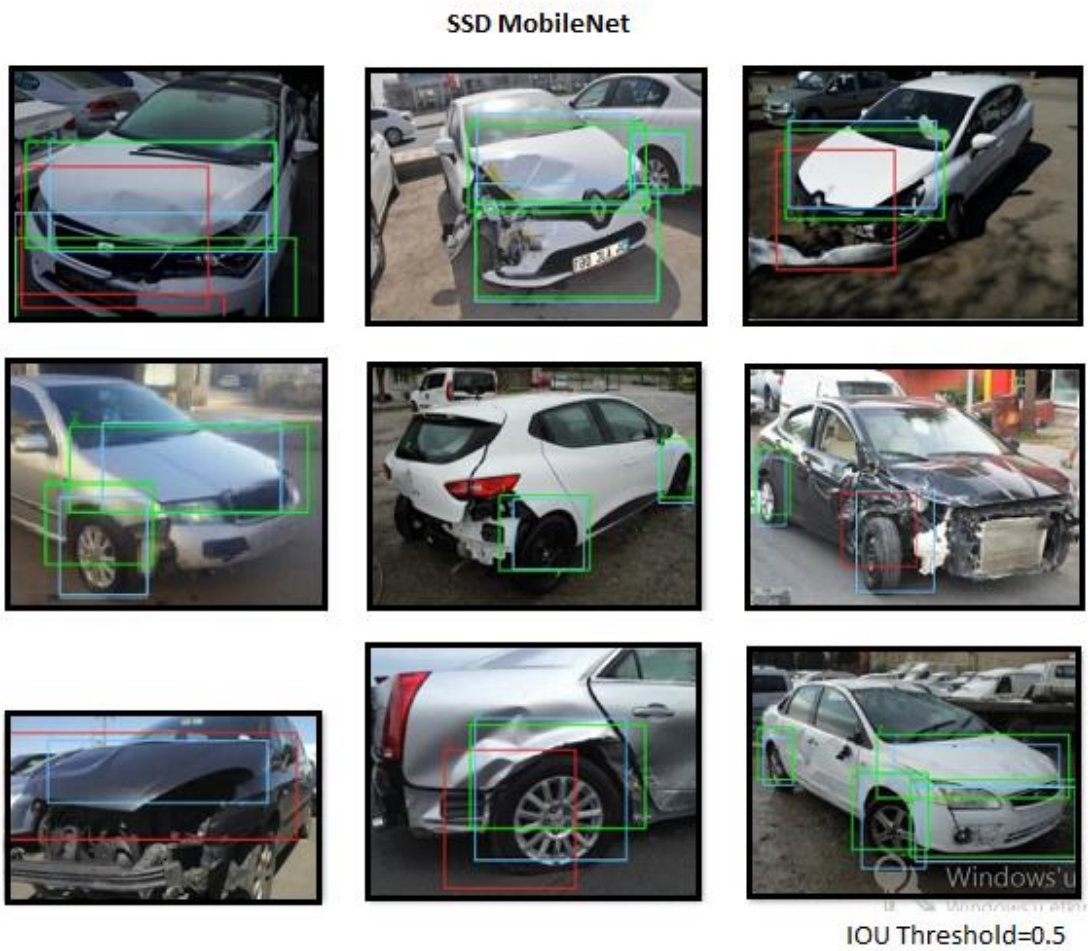

Figure 4.6 : Examples from the results of the experiment with SSD.

4.3.3. Comparison of experiments with Faster R-CNN and SSD

When we compare two experiments with CNNs, we have seen that for hood, Faster R-CNN is better for precision and recall. For bumper, Faster R-CNN is better for precision and recall. For fender, Faster R-CNN is better than SSD. In all the three damage areas Faster R-CNN gives better results than SSD for classifying the damage area of the car in terms of precision.

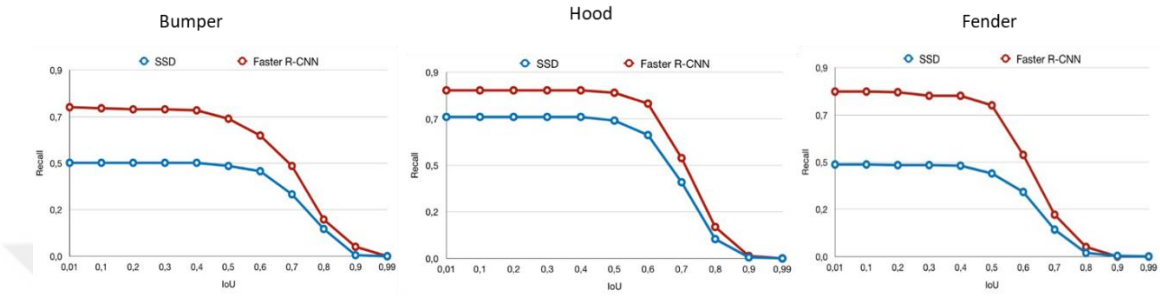

Figure 4.7 : Comparison of Experiments for object detection with Faster R-CNN and SSD.

4.4.Results For Classifying The Level Of Damage

For the classifier task of our model, we have the following results for our experiments.

4.4.1. Classifying the level of damage in our model trained with Inception v3

4.4.1.1.First Experiment with Inception v3

In our classifier model using Inception v3, we see that the model learned very fast at the first 1000 steps in the first experiment. It has a decreasing tendency. It became slower afterward.

For bumper, our model trained with Inception v3 has an accuracy rate of 0.845 with the following results.

Table 4.5 : Confusion Matrix of Classifier model for Bumper with Inception v3 – First Experiment.

		Predicted Label		
		No Damage	Minor Damage	Severe Damage
TrueLabel	No Damage	117	6	1
	Minor Damage	6	16	3
	Severe Damage	3	12	36

In Table 4.6., we see that the model is successful for no damage and severe damage classes but the minor damage classification has a precision rate of 0.47 which means more training effort is required for minor damage. We think that we should be increasing the number of images for minor car damage in our data set.

Table 4.6 : Results of Classifier model for Bumper with Inception v3 – First Experiment.

	Precision	Recall
No Damage	0.928	0.943
Minor Damage	0.470	0.640
Severe Damage	0.900	0.705

In Figure 4.8, for the first 1.000 steps, the model has learned fast but afterward accuracy has a decreasing tendency and risk of overfitting.

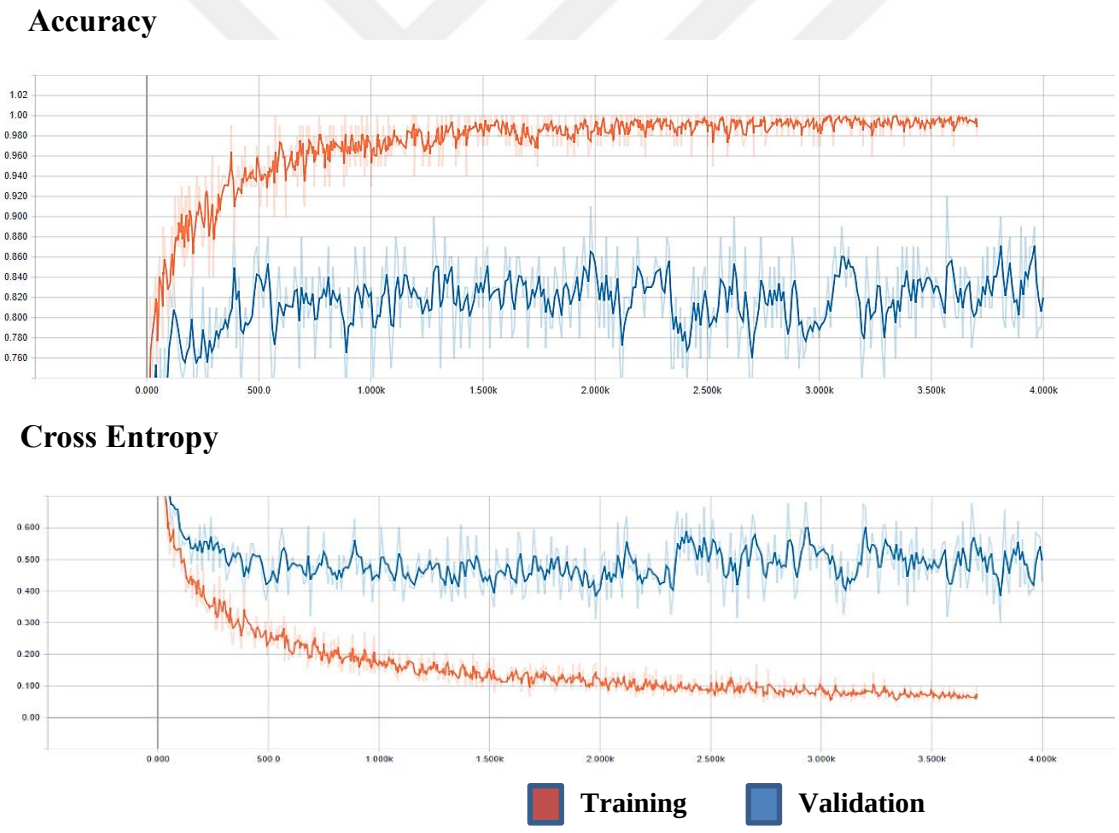

Figure 4.8 : Accuracy and Cross-Entropy graphs of Classifier model for Bumper with Inception v3 – First Experiment.

For hood, our model trained with Inception v3 has an accuracy rate of 0.795 with the following results in Table 4.7

Table 4.7 : Confusion Matrix of Classifier model for Hood with Inception v3 – First Experiment.

		Predicted Label		
		No Damage	Minor Damage	Severe Damage
True Label	No Damage	117	11	3
	Minor Damage	12	12	11
	Severe Damage	3	1	30

In Table 4.8, we see that the model is successful for no damage and severe damage classes but the minor damage classification has a precision rate of 0.5 which means more training effort is required for minor damage. We think that we should be increasing the number of images for minor damage in our data set as the model has a tendency to classify images between no damage and minor damage classes with errors.

Table 4.8 : Results of Classifier model for Hood with Inception v3 – First Experiment.

	Precision	Recall
No Damage	0.886	0.893
Minor Damage	0.500	0.342
Severe Damage	0.682	0.882

In Figure 4.9, for the first 700 steps, the model has learned fast but afterward accuracy has a decreasing tendency and risk of overfitting.

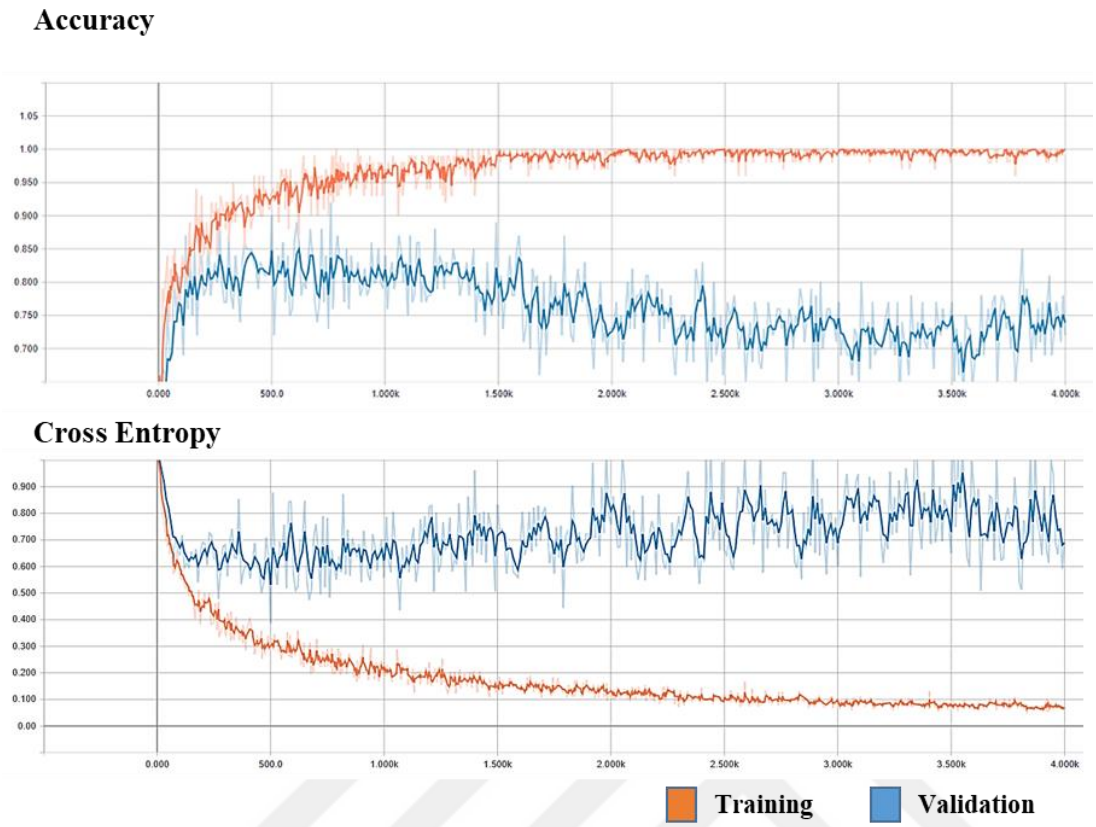

Figure 4.9 : Accuracy and Cross-Entropy graphs of Classifier model for Hood with Inception v3 – First Experiment.

For Fender, our model trained with Inception v3 has an accuracy rate of 0.816 with the following results.

Table 4.9 : Confusion Matrix of Classifier model for Fender with Inception v3 – First Experiment.

		Predicted Label		
		No Damage	Minor Damage	Severe Damage
True Label	No Damage	275	36	7
	Minor Damage	15	32	11
	Severe Damage	4	1	21

In Table 4.10, we see that the model is successful for no damage class but the minor damage classification has a precision rate of 0.464 and the severe damage classification has a precision rate of 0.538 which means more training effort is required for both classes. We think that we should be increasing the number of images for minor

and major damage images in our data set as the model has a tendency to classify images between major damage and minor damage classes with errors.

Table 4.10 : Results of Classifier model for Fender with Inception v3 – First Experiment.

	Precision	Recall
No Damage	0.935	0.865
Minor Damage	0.464	0.552
Severe Damage	0.538	0.808

In Figure 4.10, for the first 1.000 steps, the model has learned fast but afterward accuracy has a decreasing tendency and risk of overfitting.

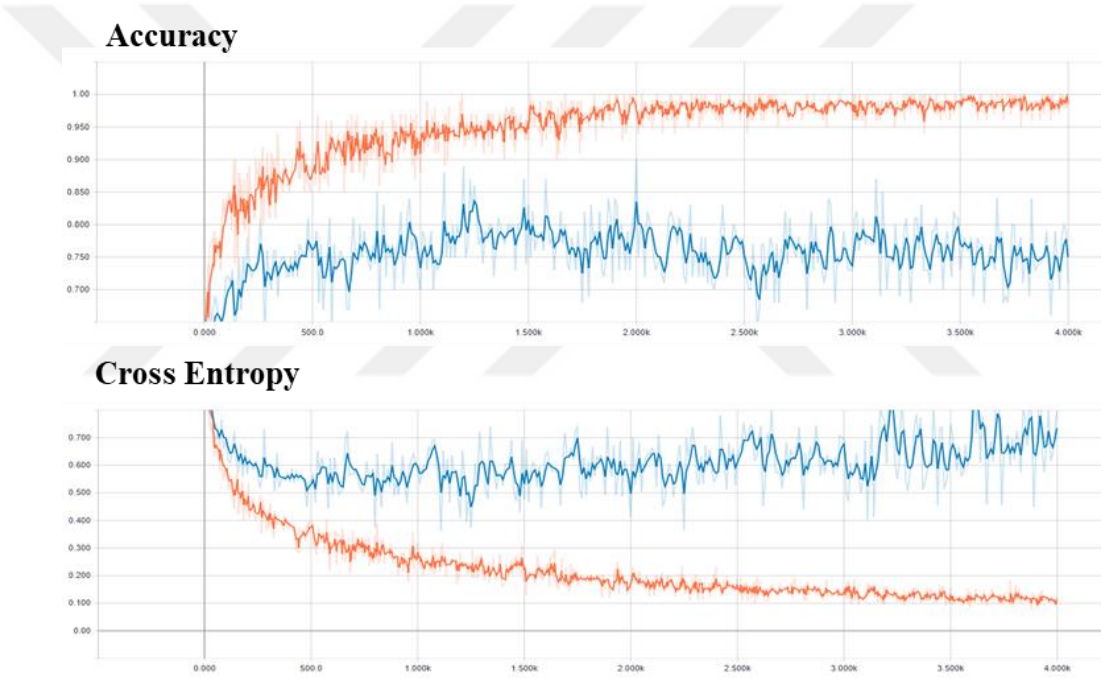

Figure 4.10 : Accuracy and Cross-Entropy graphs of Classifier model for Fender with Inception v3 – First Experiment.

Figure 4.11 : Examples from the results of the first experiment with Inception v3.

4.4.1.2. Second experiment with Inception v3

In our first experiment we understand that for minor damage classification, more training effort is required in all selected damage areas (bumper, hood, and fender). Thus we decided to add more pictures for minor damage classification in our data set and retrain our model with this new data set.

In our second experiment, we see the model learned very fast in the first 1500 steps. It has a decreasing tendency. It became slower afterward.

For bumper, our model trained with Inception v3 has an accuracy rate of 0.743 with the following results in Table 4.11.

Table 4.11 : Confusion Matrix of Classifier model for Bumper with Inception v3 – Second Experiment.

		Predicted Label		
		No Damage	Minor Damage	Severe Damage
TrueLabel	No Damage	112	7	5
	Minor Damage	8	32	25
	Severe Damage	4	20	56

In Table 4.12, we see that the model is successful for no damage and severe damage classes but the minor damage classification has precision rate of 0.542 which means our efforts of increasing the number of images for Minor damage in second experiment have limited positive effect in the results as the precision of minor damage has increased from 0.470 to 0.542.

Table 4.12 : Results of Classifier model for Bumper with Inception v3 – Second Experiment.

	Precision	Recall
No Damage	0.903	0.903
Minor Damage	0.542	0.492
Severe Damage	0.651	0.700

In Figure 4.12, for the first 1.000 steps, the model has learned fast but afterwards accuracy has a decreasing tendency and risk of overfitting.

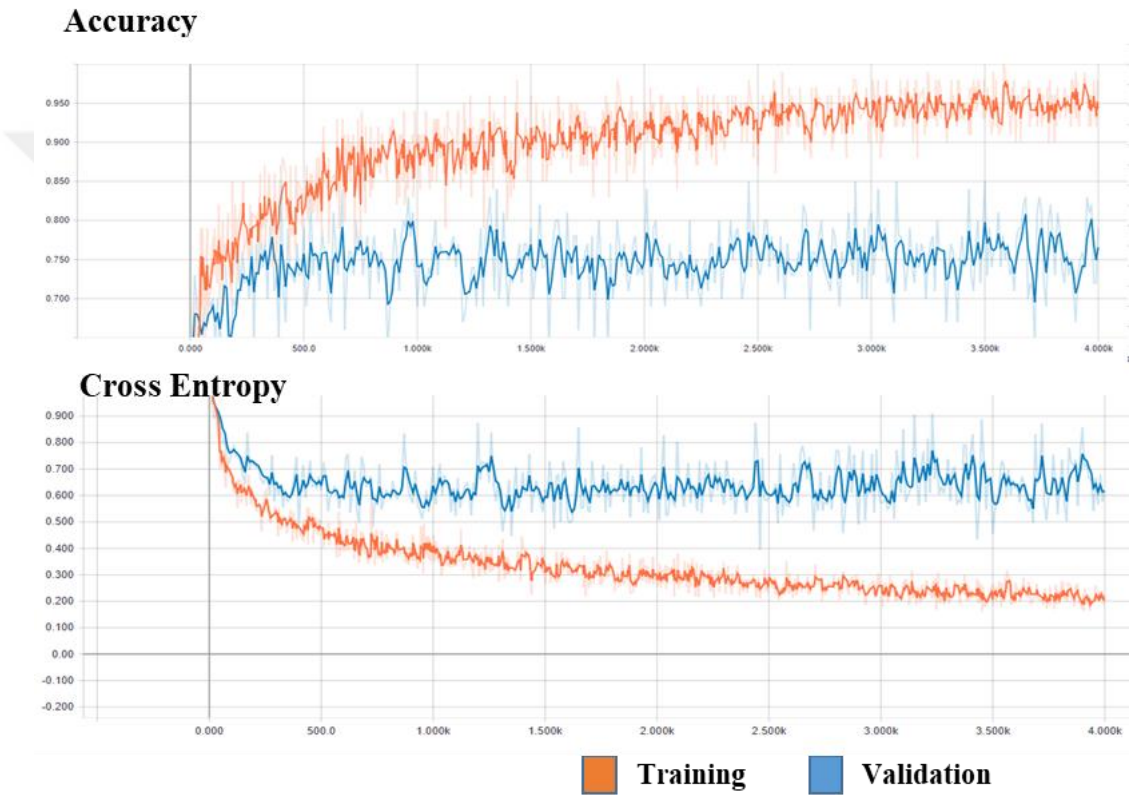

Figure 4.12 : Accuracy and Cross-Entropy graphs of Classifier model for Bumper with Inception v3 – Second Experiment.

For Hood, in the second experiment, our model trained with Inception v3 has an accuracy rate of 0.644 with the following results in Table 4.13.

Table 4.13 : Confusion Matrix of Classifier model for Hood with Inception v3 – Second Experiment.

		Predicted Label		
		No Damage	Minor Damage	Severe Damage
True Label	No Damage	117	10	4
	Minor Damage	4	20	56
	Severe Damage	8	18	44

In Table 4.14, we see that the model is successful for no damage class but precision for the minor damage classification and the severe damage classification has decreased from 0.500 to 0.416 and 0.682 to 0.423 respectively which means our efforts of increasing the number of images for Minor damage in the second experiment have a negative impact on the results.

Table 4.14 : Results of Classifier model for Hood with Inception v3 – Second Experiment.

	Precision	Recall
No Damage	0.906	0.893
Minor Damage	0.416	0.250
Severe Damage	0.423	0.628

In Figure 4.13, for the first 700 steps, the model has learned fast but afterward accuracy has a decreasing tendency and risk of overfitting.

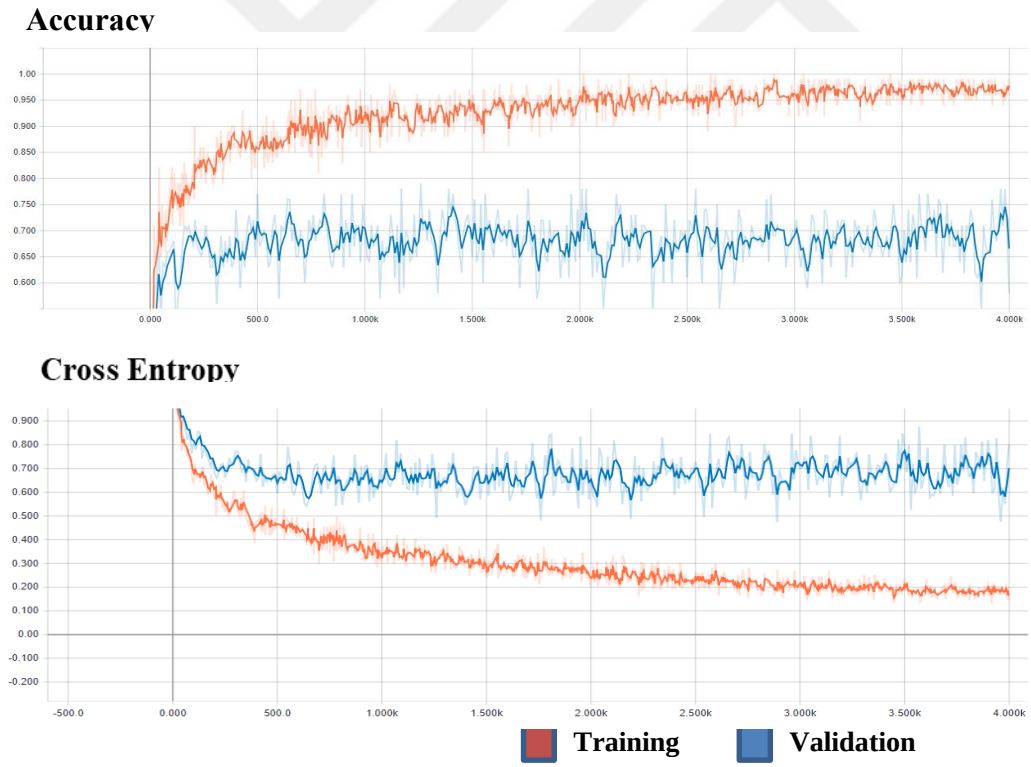

Figure 4.13 : Accuracy and Cross Entropy graphs of Classifier model for Hood with Inception v3 – Second Experiment.

For Fender, in the second experiment, our model trained with Inception v3 has an accuracy rate of 0.751 with the following results in Table 4.15

Table 4.15 : Confusion Matrix of Classifier model for Fender with Inception v3 – Second Experiment.

		Predicted Label		
		No Damage	Minor Damage	Severe Damage
True Label	No Damage	273	37	7
	Minor Damage	12	43	29
	Severe Damage	5	26	34

In Table 4.16, we see that the model is successful for no damage class but precision for the minor damage classification and the severe damage classification has decreased from 0.464 to 0.405 and 0.538 to 0.485 respectively which means our efforts of increasing the number of images for Minor damage in the second experiment have a negative impact on the results.

Table 4.16 : Results of Classifier model for Fender with Inception v3- Second Experiment.

	Precision	Recall
No Damage	0.941	0.861
Minor Damage	0.405	0.512
Severe Damage	0.485	0.523

In Figure 4.14, for the first 1.200 steps, the model has learned fast but afterward accuracy has a decreasing tendency and risk of overfitting.

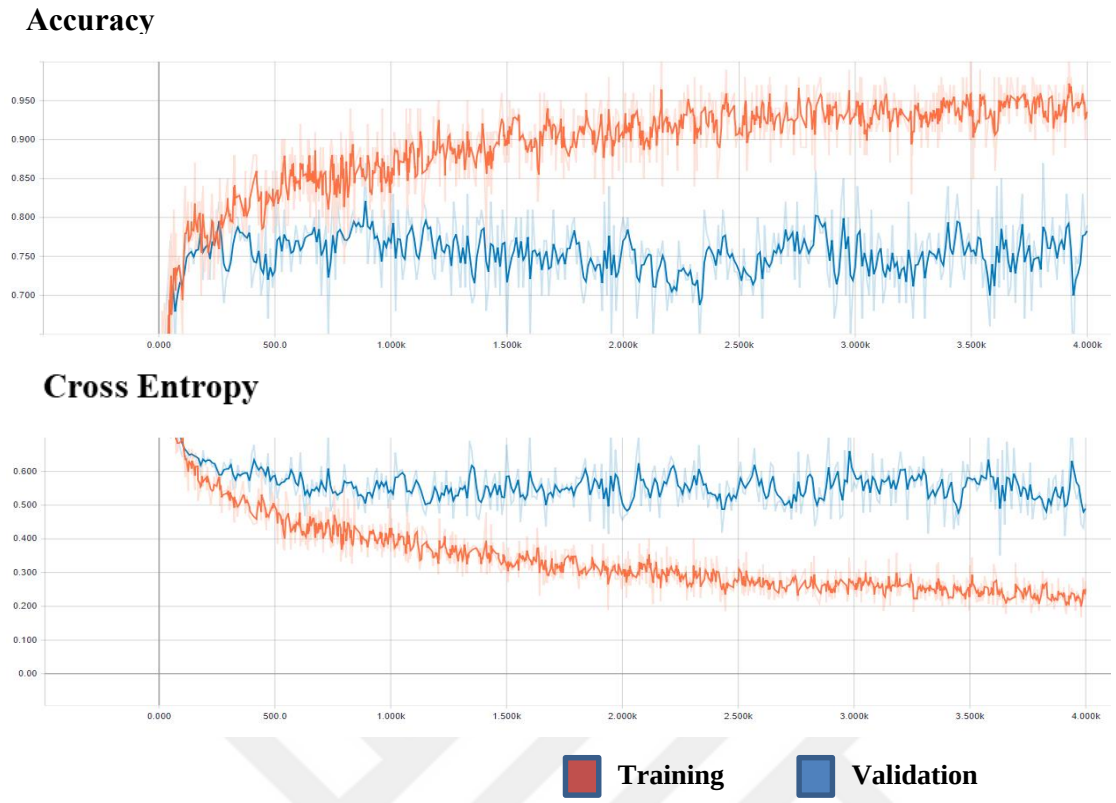

Figure 4.14 : Accuracy and Cross-Entropy graphs of Classifier model for Fender with Inception v3 – Second Experiment.

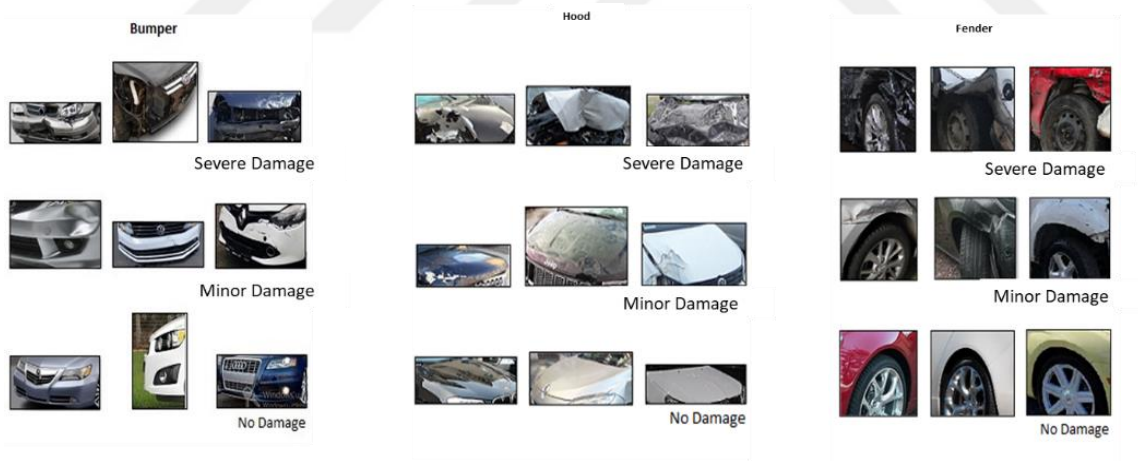

Figure 4.15 : Examples from the results of the second experiment with Inception v3.

4.4.1.3. Comparison of both experiments with Inception v3

Our aim for doing the second experiment was to increase the accuracy of minor damage class in our model with inception v3, and we could not observe the expected results when we compare both experiments. We see that Experiment 1 has better results in terms of precision except for minor damage classifier for the bumper, no damage classifier for the hood and no damage classifier for the fender.

Table 4.17 : Comparison of Both Experiments with Inception v3

	Bumper				**Hood**				**Fender**			
	Experiment 1		Experiment 2		Experiment 1		Experiment 2		Experiment 1		Experiment 2	
	Prec.	Reca.	Prec.	Reca.	Prec.	Reca.	Prec.	Reca.	Prec.	Reca.	Prec.	Reca.
ND	**0.929**	0.944	0.903	0.903	0.886	0.893	**0.906**	0.893	0.935	0.865	**0.941**	0.861
MD	0.471	0.640	**0.542**	0.492	**0.500**	0.343	0.416	0.250	**0.464**	0.552	0.405	0.512
SD	**0.900**	0.706	0.651	0.700	**0.682**	0.706	0.423	0.628	**0.538**	0.808	0.485	0.523

4.4.2. Classifying the level of damage in our model trained with MobileNet

4.4.2.1.First experiment with MobileNet

In our classifier model using MobileNet, we see the model learned very fast in the first 500 steps for the first experiment. It has a decreasing tendency. It became slower afterward.

For Bumper, our model trained with MobileNet has an accuracy rate of 0.77 with the following results in Table 4.18. We see that the learning rate is very limited after 400 steps.

Table 4.18 : Confusion Matrix of Classifier model for Bumper with MobileNet – First Experiment.

		Predicted Label		
		No Damage	Minor Damage	Severe Damage
True Label	No Damage	122	2	0
	Minor Damage	10	12	3
	Severe Damage	6	25	20

In Table 4.19., we see that the model is successful for no damage and severe damage classes but the minor damage classification has a precision rate of 0.308 which means more training effort is required for minor damage. We think that we should be increasing the number of images for minor car damage in our data set.

Table 4.19 : Results of Classifier model for Bumper with MobileNet – First Experiment.

	Precision	**Recall**
No Damage	0.884	0.984
Minor Damage	0.308	0.480
Severe Damage	0.869	0.392

In Figure 4.16, for the first 400 steps, the model has learned fast but afterward accuracy has a decreasing tendency and risk of overfitting.

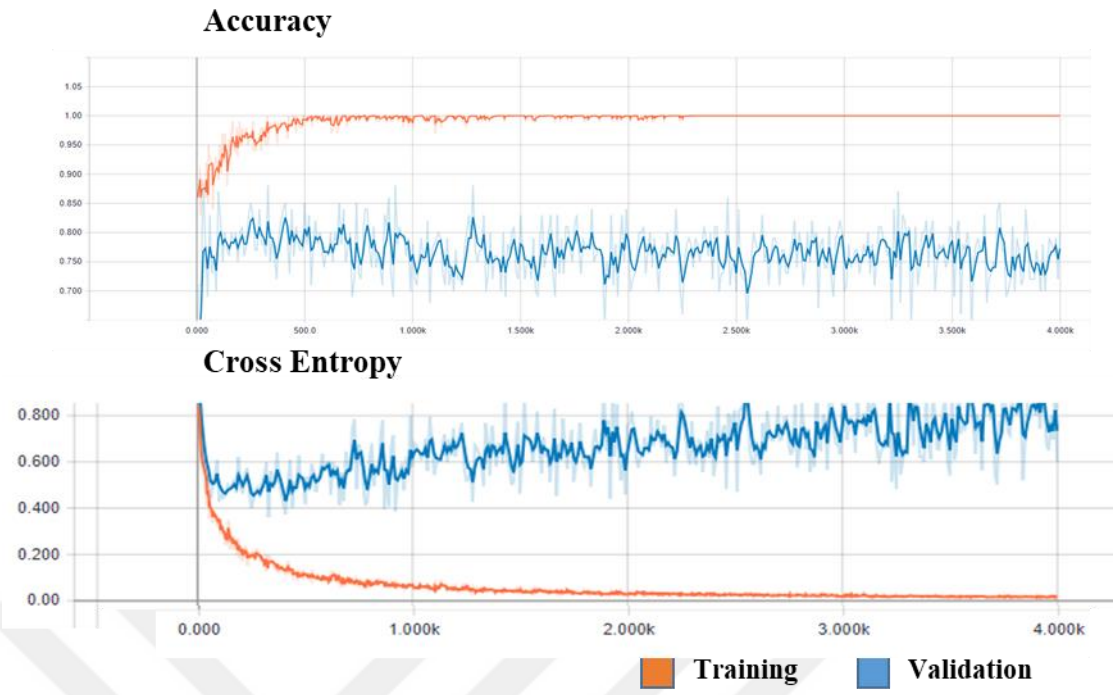

Figure 4.16 : Accuracy and Cross-Entropy graphs of Classifier model for Bumper with MobileNet – First Experiment.

For Hood, our model trained with MobileNet has an accuracy rate of 0.845 with the following results in Table 4.20. We see that the learning rate is very limited after 450 steps.

Table 4.20 : Confusion Matrix of Classifier model for Hood with MobileNet – First Experiment.

		Predicted Label		
		No Damage	Minor Damage	Severe Damage
True Label	No Damage	123	7	1
	Minor Damage	10	19	6
	Severe Damage	3	4	27

In Table 4.21, we see that the model is successful for no damage and severe damage classes but the minor damage classification has a precision rate of 0.633 which means more training effort is required for minor damage. We think that we should be increasing the number of images for minor damage in our data set as the model has a tendency to classify images between no damage and minor damage classes with errors.

Table 4.21 : Results of Classifier model for Hood with MobileNet – First Experiment.

	Precision	Recall
No Damage	0.904	0.939
Minor Damage	0.633	0.543
Severe Damage	0.794	0.794

In Figure 4.17, for the first 400 steps, the model has learned fast but afterwards accuracy has a decreasing tendency and risk of overfitting.

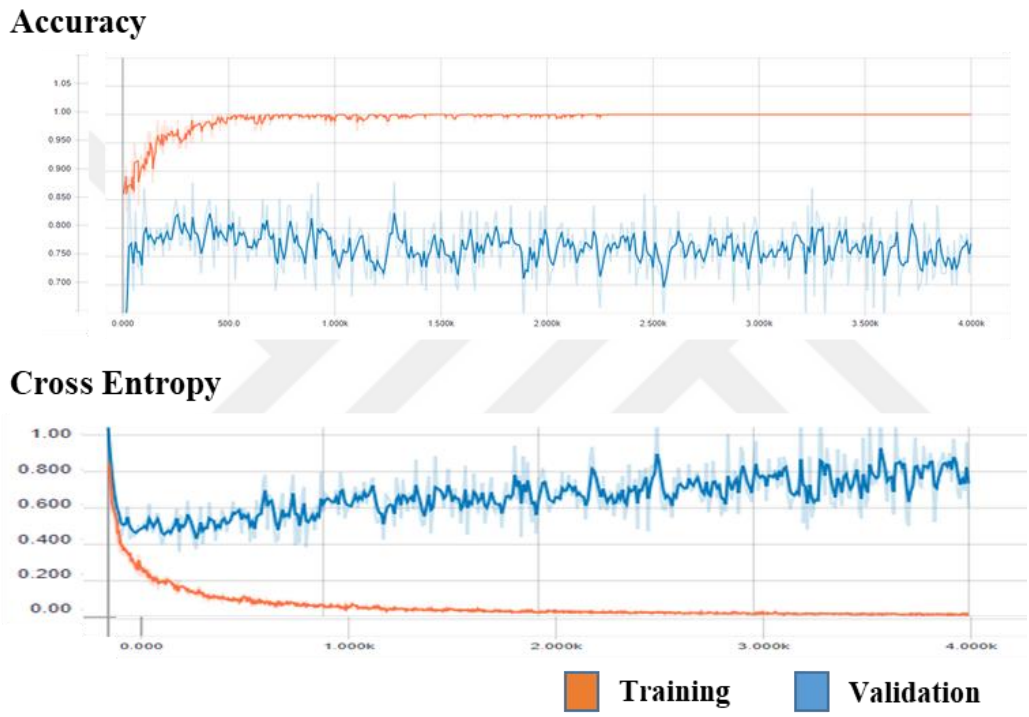

Figure 4.17 : Accuracy and Cross-Entropy graphs of Classifier model for Hood with MobileNet – First Experiment.

For Fender, our model trained with MobileNet has an accuracy rate of 0.853 with the following results in Table 4.22. We see that the learning rate is very limited after 400 steps.

Table 4.22 : Confusion Matrix of Classifier model for Fender with MobileNet – First Experiment.

		Predicted Label		
		No Damage	Minor Damage	Severe Damage
True Label	No Damage	286	25	7
	Minor Damage	14	37	7
	Severe Damage	3	3	20

In Table 4.23, we see that the model is successful for no damage class but precision for the minor damage classification and the severe damage classification are 0.569 and 0.588 respectively. We should be increasing the number of images for minor damage in our data set as the model has a tendency to classify images between no damage and minor damage classes with errors.

Table 4.23 : Results of Classifier model for Fender with MobileNet – First Experiment.

	Precision	Recall
No Damage	0.899	0.944
Minor Damage	0.569	0.638
Severe Damage	0.588	0.769

In Figure 4.18, for the first 400 steps, the model has learned fast but afterward accuracy has a decreasing tendency and risk of overfitting.

Figure 4.18 : Accuracy and Cross-Entropy graphs of Classifier model for Hood with MobileNet. – First Experiment

Figure 4.19 : Examples from the results of the experiment with the MobileNet – First Experiment.

4.4.2.2.Second experiment with MobileNet

In our first experiment we understand that for minor damage classification, more training effort is required in all selected damage areas (bumper, hood, and fender). Thus we decided to add more pictures for minor damage classification in our data set and retrain our model with this new data set.

In our classifier model using MobileNet, we see the model learned very fast in the first 500 steps for the second experiment. It has a decreasing tendency. It became slower afterward. For Bumper, our model trained with MobileNet has an accuracy rate of 0.743 with the following results in Table 4.24.

Table 4.24 : Confusion Matrix of Classifier model for Bumper with MobileNet – Second Experiment.

		Predicted Label		
		No Damage	Minor Damage	Severe Damage
True Label	No Damage	114	6	4
	Minor Damage	7	34	24
	Severe Damage	4	24	52

In Table 4.25, we see that the model is successful for no damage. Severe damage class has a precision of .650 which decreased from 0.869 in our first experiment. The minor damage classification has a precision rate of 0.531 which has increased from 0.308 in our first experiment. Our efforts for the second experiment has caused errors between Minor Damage and Severe Damage classes.

Table 4.25 : Results of Classifier model for Bumper with MobileNet – Second Experiment.

	Precision	Recall
No Damage	0.912	0.920
Minor Damage	0.531	0.523
Severe Damage	0.650	0.650

In Figure 4.20, for the first 500 steps, the model has learned fast but afterward accuracy has a decreasing tendency and risk of overfitting.

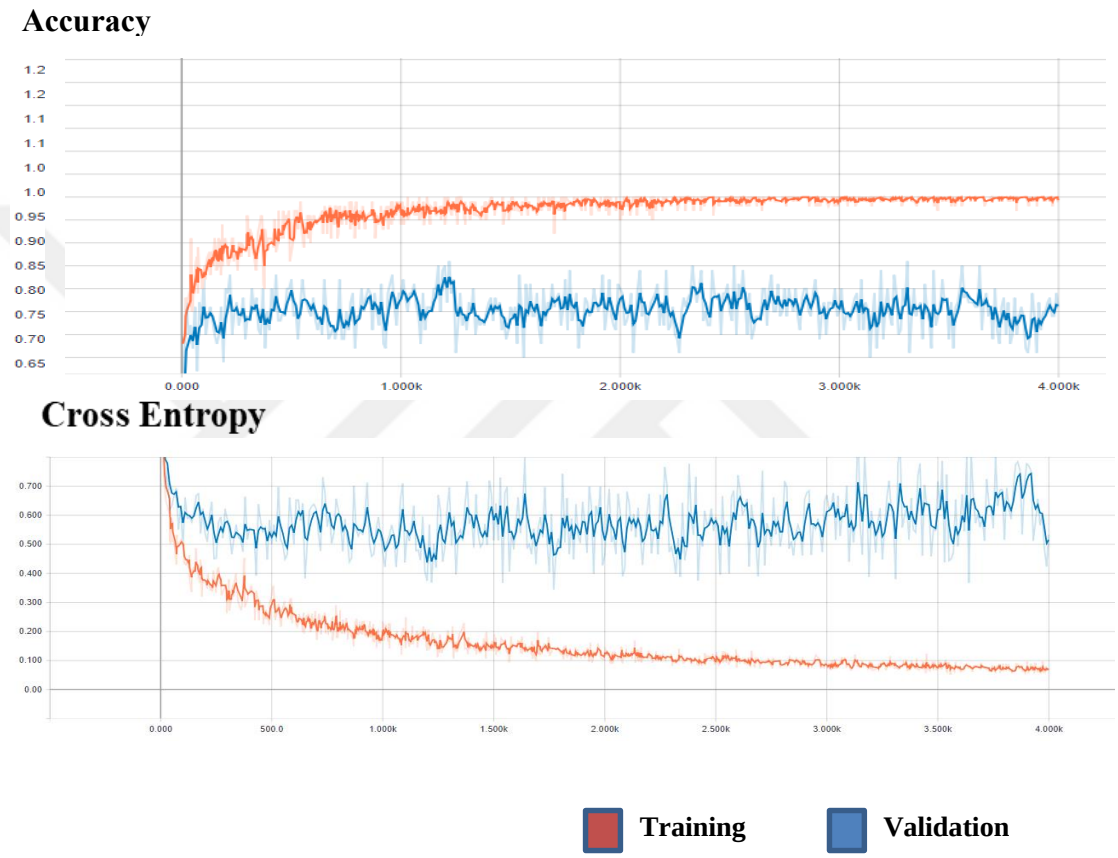

Figure 4.20 : Accuracy and Cross-Entropy graphs of Classifier model for Bumper with MobileNet – Second Experiment.

For Hood, our model trained with MobileNet has an accuracy rate of 0.756 with the following results in Table 4.26. We see that the learning rate is very limited after 500 steps.

Table 4.26 : Confusion Matrix of Classifier model for Hood with MobileNet – Second Experiment.

		Predicted Label		
		No Damage	Minor Damage	Severe Damage
True Label	No Damage	117	12	2
	Minor Damage	9	37	24
	Severe Damage	4	15	51

In Table 4.27, we see that the model is successful for no damage. But the precision of all three classes decreased Our efforts for the second experiment has caused errors between Minor Damage and Severe Damage classes.

Table 4.27 : Results of Classifier model for Hood with MobileNet – Second Experiment.

	Precision	Recall
No Damage	0.900	0.893
Minor Damage	0.578	0.528
Severe Damage	0.662	0.728

In Figure 4.21, for the first 500 steps, the model has learned fast but afterward accuracy has a decreasing tendency and risk of overfitting.

Figure 4.21 : Accuracy and Cross-Entropy graphs of Classifier model for Hood with MobileNet – Second Experiment.

For Fender, our model trained with MobileNet has an accuracy rate of 0.790 with the following results in Table 4.28. We see that the learning rate is very limited after 500 steps.

Table 4.28 : Confusion Matrix of Classifier model for Fender with MobileNet – Second Experiment.

		Predicted Label		
		No Damage	Minor Damage	Severe Damage
True Label	No Damage	286	27	5
	Minor Damage	13	48	23
	Severe Damage	2	28	35

In Table 4.29, we see that the model is successful for no damage. Severe damage class has a precision of .555 which decreased from 0.588 in our first experiment. The minor damage classification has a precision rate of 0.466 which has decreased from 0.569 in our first experiment. Our efforts for the second experiment have caused errors between Minor Damage and Severe Damage classes.

Table 4.29 : Results of Classifier model for Fender with MobileNet – Second Experiment.

	Precision	**Recall**
No Damage	0.950	0.899
Minor Damage	0.466	0.571
Severe Damage	0.555	0.538

In Figure 4.22, for the first 500 steps, the model has learned fast but afterward acuracy has a decreasing tendency and risk of overfitting.

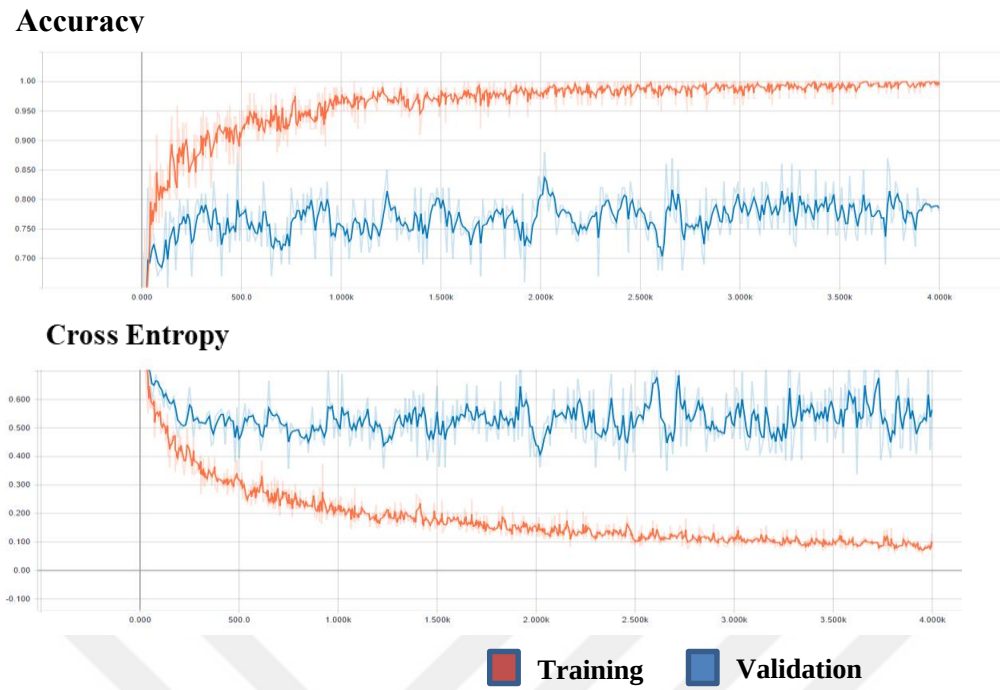

Figure 4.22 : Accuracy and Cross-Entropy graphs of Classifier model for Fender with MobileNet. – Second Experiment

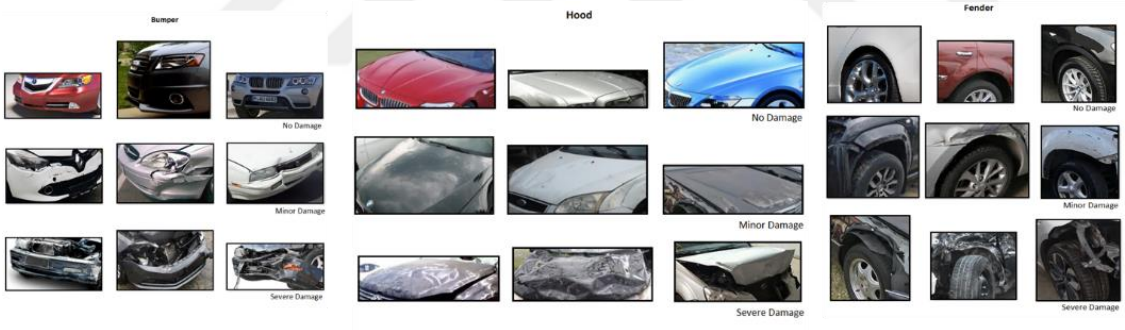

Figure 4.23 : Examples from the results of the experiment with the MobileNet – First Experiment.

4.4.2.3. Comparison of both experiments with MobileNet

Our aim for doing the second experiment was to increase the accuracy of minor damage class in our model with MobileNet, and we could not observe the expected results when we compare both experiments. We see that Experiment 2 has better results in terms of precision except for no damage classifier for the bumper, minor damage classifier for the bumper and no damage classifier for the fender.

Table 4.30 : Comparison of Both Experiments withMobileNET

	Bumper				**Hood**				**Fender**			
	Experiment 1		Experiment 2		Experiment 1		Experiment 2		Experiment 1		Experiment 2	
	Prec.	Reca.	Prec.	Reca.	Prec.	Reca.	Prec.	Reca.	Prec.	Reca.	Prec.	Reca.
ND	0.884	0.984	**0.912**	0.920	**0.904**	0.939	0.900	0.893	0.899	0.944	**0.950**	0.899
MD	0.308	0.480	**0.531**	0.523	**0.633**	0.543	0.578	0.528	**0.569**	0.638	0.466	0.571
SD	**0.869**	0.392	0.650	0.650	**0.794**	0.794	0.662	0.728	**0.588**	0.769	0.555	0.538

4.4.3. Comparison of experiments with Inception v3 and MobileNet

When we compare two experiments for damage level classification with different base CNNs, we understood that for Bumper Inception v3 has better performance, while for Hood both models have very similar results but model based on MobileNet has a slight advantage and for fender model based on MobileNet has better performance in terms of precision. We have seen that both models need better training for minor damage classification in all the damage areas.

Table 4.31 : Comparison of Experiments with Inception v3 and MobileNet.

	Bumper				**Hood**				**Fender**			
	Inception V3		MobileNet		Inception V3		MobileNet		Inception V3		MobileNet	
	Prec.	Reca.	Prec.	Reca.	Prec.	Reca.	Prec.	Reca.	Prec.	Reca.	Prec.	Reca.
ND	**.929**	.944	.884	.984	.886	.893	**.904**	.939	**.935**	.865	.899	.944
MD	**.471**	.640	.308	.480	.500	.343	**.633**	.543	.464	.552	**.569**	.638
SD	**.900**	.706	.870	.392	.682	.706	**.794**	.794	.538	.808	**.588**	.769

5. CONCLUSION

We have taken some good first steps for the application of CNNs in the context of car damage analysis for the insurance market. Detecting the area of damage for the car is a good application of object detection models. Yet, it can still be considered as a difficult task as pre-trained models are usually not specifically trained for detecting the area of damage for the car. We compared two different models trained on two different meta-architectures for our transfer learning approach.

We see that our model trained with Faster R-CNN with Inception v2 feature extractor performs better for the same data set compared to our model trained with SSD with MobileNet feature extractor. We understand that we need to perform more experiments for fender detection to get more accurate results. The second step of our approach is to classify the damage level of the selected areas we have gathered from our damage area detection models. We see that for the bumper, Inception v3 has better performance, while for Hood both models have very similar results with MobileNet's slight advantage and for the fender, model based on MobileNet performs better performance in terms of precision. We understand that for minor damage classification, more training effort is required for both models in all selected damage areas (bumper, hood, and fender). In order to understand the effect of using more images in training data, we perform a second experiment for our damage level classification model. In this experiment, we have included additional data especially for the minor damage and severe damage classes for our training efforts. Indeed, we have a slight improvement in some of the classifiers but in the second experiment, we faced with the wrong classification between severe damage and minor damage classes.

For future studies, when using transfer learning, it may be interesting to experiment with other model architectures and optimization algorithms for different use-cases in car damage analysis for the insurance market. As the pressure for more efficient operation in terms of costs increases in the markets like insurance, the players in the market have the option to use AI based automation solutions to create efficiency, decrease costs and minimize the time to serve to customers. We think that our case,

detecting the level of damage for car insurance has provided a good basis for the future studies that will focus on automating the process between the insurance company, the customer and the service company. We can create a number of different solutions even for this specific process in the insurance market.

REFERENCES

Batista, G. E., Prati, R., & Monard, M. C. (2004). A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD explorations newsletter, 6*(1), 20-29.

Camastra, F., & Vinciarelli, A. (2015). *Machine learning for audio, image and video analysis: theory and applications.* Springer.

Chawla, N. V. (2009). Data mining for imbalanced datasets: An overview. In *Data mining and knowledge discovery handbook* (pp. 875-886). Springer.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., & Kuksa, P. (2011). Natural language processing (almost) from scratch. *Journal of machine learning research, 12*(Aug), 2493-2537.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems, 2*, 303-314.

Dai, J., Li, Y., He, K., & Sun, J. (2016). R-FCN: Object Detection via Region-based Fully Convolutional Networks. *Advances in Neural Information Processing Systems 29 (NIPS 2016)*, (pp. 379-387).

de Deijn, J. (2018). Automatic Car Damage Recognition using Convolutional Neural Networks.

Domingos, P. M. (2012). A few useful things to know about machine learning. *Commun. acm, 55*(10), 78-87.

Everingham, M., Van Gool, L., Wiliams, C. K., Winn, J., & Zisserman, A. (2010). The PASCAL Visual Object Classes (VOC) Challenge. *International journal of computer vision*, 303-338.

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. *IEEE conference on computer vision and pattern recognition*, (pp. 580 -587).

Girshick., R. (2015). Fast R-CNN. *IEEE International Conference oncomputer vision*, (pp. 1440–1448).

Graves, A., Mohamed, A.-r., & Hinton, G. (2013). Speech recognition with deep recurrent neural networks. *2013 IEEE international conference on acoustics, speech and signal processing* (pp. 6645-6649). IEEE.

Hastie, T., Tibshirani, R., & Friedman, J. (2011). The elements of statistical learning: data mining, inference, and prediction, Springer Series in Statistics. Springer.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *IEEE conference on computer vision and pattern recognition*, (pp. 770-778).

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., . . . Adam, H. (2017). *Mobilenets: Efficient convolutional neural networks for mobile vision applications.* Retrieved from arXiv.org: https://arxiv.org/abs/1704.04861

Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., . . . Guadarrama, S. (2017). Speed/accuracy trade-offs for modern convolutional object detectors. *IEEE Conference on Computer Vision and Pattern Recognition* (pp. 7310-7311). Google research.

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *International Conference on Machine Learning*, (pp. 448-456).

Jayawardena, S. (2013). Image Based Automatic Vehicle Damage Detection.

Jordan, M., & Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. *Science, 249*, 255-260.

Krizhevsky, A., Sutskever, I., & Hinton, G. (2012). *Imagenet classification with deep convolutional neural networks.*

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation*, 541-551.

Li, P., Shen, B., & Dong, W. (2018). An Anti-fraud System for Car Insurance Claim Based on Visual Evidence. *arXiv preprint arXiv:1804.11207*.

Liu, W. A. (2016). Ssd: Single shot multibox detector. *European Conference on computer vision* (pp. 21-37). Springer.

Murphy, K. (2012). *Machine learning: a probabilistic perspective.* MIT Press.

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. *Proceedings of the 27th international conference on machine learning (ICML-10)*, (pp. 807-814).

Patil, K., Kulkarni, M., Sriraman, A., & Karande, S. (2017). Deep learning based car damage classification. *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)* (pp. 50-54). IEEE.

Ramachandran, P., Zoph, B., & Le, Q. (2017). Swish: a self-gated activation function. *arXiv preprint arXiv:1710.05941, 7*.

Ren, S., He, K., Girshick, R., & Sun, J. (2016). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *Advances in neural information processing systems*, 911-99.

Ripley, B. D. (2007). *Pattern recognition and neural networks.* Cambridge university press.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., . . . Berg, A. C. (2015). Imagenet large scale visual recognition challenge. International

Journal of Computer Vision 115(3), 211–252. *International Journal of computer vision*, 211-252.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, 85-117.

Simonyan, K., & Zisserman, A. (2014). *Very deep convolutional networks for large-scale image recognition*. Retrieved from arXiv: arXiv:1409.1556

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., . . . Rabinovich, A. (2014). Going deeper with convolutions. *IEEE Conference on computer vision and pattern recognition*, (pp. 1-9).

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2015). Rethinking the Inception Architecture for Computer Vision.

Zeiler, M. D. and Fergus, R. (2014). Visualizing and understanding convolutional networks. *European conference on computer vision* (pp. 818–833). Springer.

APPENDICES

Transfer learning based on Faster R-CNN

For training, we used the following configuration settings for Faster R-CNN in our experiment:
Meta Architecture: Faster R-CNN
Feature Extractor: Inception v2
Batch size: 1
Activation Function: RELU
Optimizer: Momentum optimizer
Learning rate : 0.0002 (900000 step)
0.000002 (1200000 step)
Momentum Optimizer Value : 0.9
Grid Anchor Generator
Scales: [0.25, 0.5, 1.0, 2.0]
aspect_ratios: [0.5, 1.0, 2.0]
height_stride : 16
width_stride : 16

First stage post processing
Nms IoU threshold : 0.7
Maximum proposals : 300
Localization loss weight : 2.0
Objectness loss weight : 1.0
Initial crop size : 14
Maxpool kernel size : 2
Maxpool stride : 2
Second stage post processing
IoU_threshold : 0.6
max_detections_per_class : 100
max_total_detections : 300
second_stage_localization_loss_weight: 2.0
second_stage_classification_loss_weight: 1.0
Gradient clipping by normalization: 10.0

Transfer Learning Based on SSD

For training, we used the following configuration settings for our model based on SSD.

Meta Architecture: SSD
Feature Extractor: Mobile.net
Batch size: 24

Activation Fonction: RELU 6
Optimizer: RMSprop optimizer
Learning rate: 0.0004
    Decay steps: 800720
    Decay factor: 0.95
Momentum Optimizer Value: 0.9
Ssd Anchor Generator
    num_layers: 6
    min_scale: 0.2
    max_scale: 0.95
    aspect_ratios: 1.0
    aspect_ratios: 2.0
    aspect_ratios: 0.5
    aspect_ratios: 3.0
    aspect_ratios: 0.33

CURRICULUM VITAE

Name Surname **: Cihat Tolga ARTAN**

Place and Date of Birth **: Adapazarı, 23.02.1974**

E-Mail **: tartan@luckyeye.com**

EDUCATION **:**

- **B.Sc.** **:** 1992, Istanbul Technical University, Faculty of Management, Industrial Engineering Department, Industrial Engineering Programme

PROFESSIONAL EXPERIENCE AND REWARDS:

- 1996-2000 The Media Group, Business Development Manager
- 2000 - Present LuckyEye, CEO

PUBLICATIONS, PRESENTATIONS AND PATENTS ON THE THESIS:

- **Artan C.T. and Kaya T.** 2019: INFUS 2019 – International Conference on Intelligent and Fuzzy Systems. July 23-25, 2019 Istanbul, Turkey. (Presentation Instance)