UNIVERSITY OF TARTU

Institute of Computer Science
Computer Science Curriculum

**Navid Bamdad Roshan**

# Change Detection in HD-Maps Using Camera Images for Autonomous Driving

**Master's Thesis (30 ECTS)**

Supervisors:

Dmytro Fishman, MSc
Naveed Muhammad, PhD

Tartu 2021

# Change Detection in HD-Maps Using Camera Images for Autonomous Driving


## Abstract:

Self-driving vehicles have been an exciting field of research for both industry and academia in the last decade. The map is one of the challenging aspects of this research. It has been centuries that maps are used in transportation for routing. Accordingly, autonomous vehicles can use maps for routing as well. Some maps that can be used for autonomous driving are called high-definition (HD) maps. HD maps are more accurate than ordinary maps. Details of the HD maps are in centimeter-level accuracy. They also have more details compared to regular maps. For instance, HD-maps have information about the surrounding environment of the autonomous vehicle like details about streets, lanes, traffic rules, traffic signs, traffic lights, etc. This additional information assists autonomous vehicles in perceiving the environment better to move safer and more efficiently. Thus, autonomous vehicles need up-to-date details in HD maps all the time. So, in case of any changes in the environment, the changes must be detected, and HD maps must be updated accordingly. Therefore, it is essential to design an automatic solution for detecting the changes in the environment. This work proposes an automatic streets' drivable area change detection pipeline. The proposed pipeline detects any changes that alter the drivable path of the streets. The detected changes can be used to update the HD maps later.



**Keywords:**
Neural Networks, HD maps, change detection, drivable area change detection

**CERCS:**
P170 (Computer science, numerical analysis, systems, control)

# Kaamera piltide abil muutuste tuvastamine isejuhtivatele sõidukitele mõeldud täppiskaartidel.

**Lühikokkuvõte:**

Isejuhtivad sõidukid on viimase kümnendi jooksul olnud põnev uurimisvaldkond nii tööstuses kui ka akadeemilises ringkonnas. Läbi sajandite on kaardid olnud abiks reisimisel ja teekonna planeerimisel. Isejuhtivad sõidukid saavad samuti kasutada kaarte oma teekonna planeerimiseks. Kaarte, mis võimaldavad autode isejuhtimist nimetatakse täppsikaartideks ja need on käesoleva uurimistöö keskseks elemendiks Täppiskaartide täpsust saab mõõta sentimeetrites, mis teeb nad tavakaartidest tunduvalt täpsemaks, lisaks on nende andmed tunduvalt detailsemad. Näiteks on täppiskaardil info isejuhtivat sõidukit ümbritseva keskkonna kohta: andmed tänavate, teede, liikluseeskirja, liiklusmärkide, tänavavalgustuspostide jms kohta. See täiendav teave aitab isejuhtival sõidukil tajuda keskkonda paremini, liikuda turvalisemalt ja tõhusamalt. Seega, on isejuhtivatele sõidukitele oluline, et täppiskaartidel olevad andmed oleksid ajakohased. Keskkonna muutuste tuvastamine on kasulik eriti kuna seda teavet saab kasutada ka täppiskaartide uuendamiseks. Seetõttu on oluline luua automaatne lahendus keskkonna muutuste avastamiseks. Käesolev töö pakub välja automaatse töövoo muutuste tuvastamiseks tänava sõidetaval alal. See võimaldab tuvastada kõik muutused, mis mõjutavad sõidutrajektoori. Tuvastatud muutusi saab kasutada hilisemaks täppiskaartide uuendamiseks.

**Võtmesõnad:**
Närvivõrgud, täppiskaardid, muutuste tuvastamine, sõidetava ala muutuste tuvastamine

**CERCS:**
P170 (Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine)

# 1. Introduction

Transportation has been an essential aspect of human's everyday life. There have been many inventions throughout history to ease transportation, like the invention of the wheel, carriage, cars, aircraft, etc. Despite outstanding improvements in transportation, the progress continues. Autonomous driving is a very young research area that attracts the best minds from both the industry and academia [1]. Although some manufactured cars already have some levels of autonomy like automated steering, lane-keeping, or emergency stop in the presence of obstacles, the driver still needs to be in control of the vehicle [2]. However, the goal of autonomous driving is to automate the whole process of driving and eliminate the need for a human driver. This level of autonomy is referred to as level 5 [3]. The importance of autonomous driving includes but is not limited to relieving humans from driving. It will also reduce the number and casualties of vehicle accidents caused by human errors [2]. They can also reduce traffic and pollution caused by vehicles thanks to route optimization [4].

There are two main research approaches for autonomous driving currently, modular and end-to-end driving [1]. The modular approach focuses on creating a system that consists of different modules such as localization, perception, planning, and control [5]. An advantage of this approach is that the functionality of each module can be monitored and analyzed individually. Therefore, detecting the reason for bad system behavior is easier in the modular approach. The cost of creating such a system, however, is high [1]. On the other side, the end-to-end approach is an alternative solution for the modular approach. In the end-to-end approach, a neural network is responsible for outputting the steering and acceleration command by getting the sensory data as input [1]. Unlike the modular approach, the end-to-end approach usually has a simpler architecture with the cost of losing the interpretability.

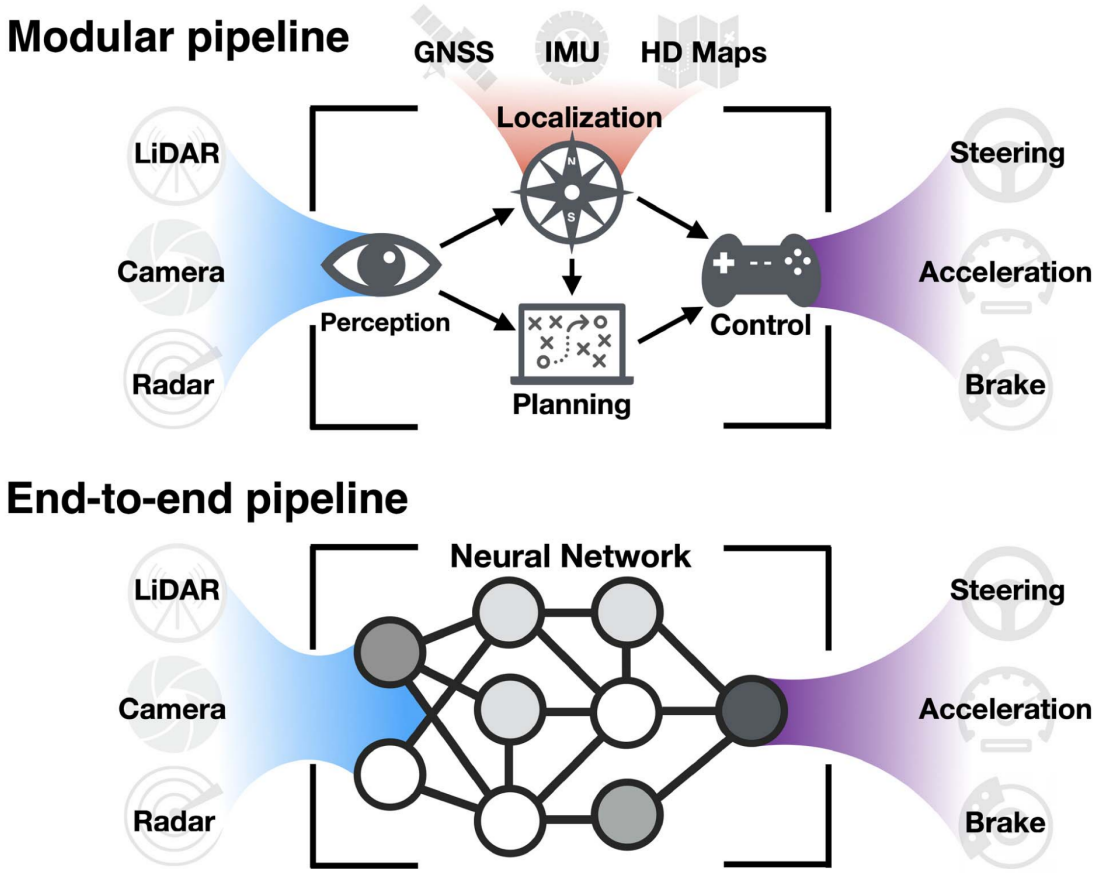



Fig 1.1. Modular and end-to-end pipelines. The modular pipeline for autonomous driving consists of many interconnected modules. The end-to-end approach treats the entire pipeline as one learnable machine learning task [1].

As discussed above, in the modular approach, the whole system is composed of different components, and each component has its own duty. One of the key technologies for autonomous driving is the high-definition (HD) map [6]. The maps that are particularly made for autonomous driving are called high-definition (HD) maps. These maps have information about the surrounding environment of the autonomous vehicle with centimeter-level precision [7]. This information consists of details about streets, lanes, traffic rules, traffic signs, traffic lights, etc. HD maps can cover two challenges in autonomous driving. First, an autonomous vehicle can localize itself relative to its environment using the precise details of the HD map. Second, the planning module can utilize the details of the HD maps about the existing traffic signs, traffic lights, streets, and their drivable area, etc, to plan the path that the autonomous vehicle must traverse. Therefore, it is essential to keep the HD maps up to date all the time. The details of the streets' drivable area are part of the vital HD map information

that must be kept up to date as the autonomous vehicle depends on to ensure safety while moving. To keep the HD maps up to date, we have to detect the changes in the environment first. Detecting the changes could be challenging and costly because the changes might be very tiny, so it is difficult to notice such changes manually. And it is very costly to monitor the streets of a city or even a region for changes.

## 1.1. Contribution

This work proposes the drivable area change detection pipeline to automate the process of detecting any change in a street's drivable area caused by roadworks, constructions, etc. The proposed pipeline uses pair camera images (new and old camera images) of a street to compare the drivable area of a street with its old drivable area. Moreover, two datasets are generated for training of the proposed drivable area change detection model.

## 1.2. Thesis Structure

An introductory description about Autonomous Driving is provided in this chapter. Few related works are discussed in chapter 2. Chapter 3 explains the datasets generation and splitting process. In chapter 4 utilized methods are elaborated. Details of the experiments designs and the results can be found in chapter 5. A discussion about the results is provided in chapter 6. And chapter 7 concludes this work.

# 2. Related Works

There are a moderate number of papers published in the narrow area of streets change detection. A few of these papers are discussed in this chapter.

In 2018 Alcantarilla et al developed a street-view change detection solution using deconvolutional networks. [8] The authors in their work, used the street-view videos captured by the vehicle-mounted monocular cameras to detect the structural changes in the street-view. They proposed a multi-sensor fusion SLAM and fast 3D reconstruction pipeline to register the image pairs. These registered image pairs are fed into the proposed deep deconvolutional neural network to perform pixel-wise change detection. The input dimension of the neural network is 240x180x6 (two RGB images), and the output is a change map of size 240x180. The resulting change map is a mask that shows the pixel-wise change in the input pair images. They introduced a new dataset for training their neural network. This new dataset is called VL-CMU-CD, and it is publicly available [8]. This work, however, cannot be directly used for detecting the changes in the drivable area of the streets as it is mostly focusing on detecting the changes caused by the objects like bins, vehicles, etc.

Another work published in 2018 by Varghese et al. [9] has presented a deep learning architecture for visual change detection called ‘ChangeNet’. They proposed a deep learning architecture capable of detecting the changes between image pairs. Their architecture included the Siamese network [38][39] first. Output of the Siamese network is fed into a fully convolutional network (FCN). The input of the Siamese network is a Fpair of images of size 224x224x3 each. The output of the network is a change map of size 224x224x11. 224x224 is the dimension of the input images, and 11 is the number of semantic classes like barriers, bins, construction, vehicles, etc. In this work, the VL-CMU-CD dataset [8] is used to train the proposed neural network. Although this work shows a good level of object-based classification, it cannot be directly used for drivable area change detection. Because this work performs a pixel-based change detection which means the images of the streets must be from the exact same point with the exact viewpoint to be able to compare the drivable areas of the streets. Moreover, this work tries to detect the changes caused by objects like vehicles, bins, etc. These objects are temporary and they do not necessarily cause a change in the drivable area of the streets.

In 2020 Vladislav Fediukov - a Master’s student at University of Tartu - proposed a method to detect the changes in HD-maps using LiDAR point clouds. [10] In this work, the scanned point clouds of the same places on different days are compared together to

detect the changes. First, the dynamic objects like cars are removed using a Point Pillar neural network. Then, the point clouds of the same places are registered using Iterative Closest Point and Coherent Point Drift algorithms. After that, the Hausdorff and averaged point-to-surface distance metrics are used to calculate the differences between two sets of point clouds. If the difference is more than a threshold, it is considered a change in the environment. This work shows a good performance in detecting the changes in the HD-maps using the LiDAR point clouds. However, LiDAR sensors are considerably more expensive than other sensors like cameras. So, detecting the changes in HD-maps using the LiDAR point clouds could be very expensive.

Considering the above mentioned papers, convolutional neural architectures have demonstrated promising outcomes in detecting the changes using camera images. However, current works mostly focus on detecting any changes in street-view images. So, these works cannot be directly used for streets' drivable area change detection. Vladislav Fediukov [10] used LiDAR point clouds to detect the changes in the drivable area of the streets. However, it is not very cost efficient to use LiDAR sensors. We propose streets' drivable area change detection pipeline in this work. The proposed pipeline uses the camera images to detect the changes in drivable path of the roads.

# 3. Dataset

## 3.1. Oxford RobotCar Dataset

To the best of our knowledge, at the time of writing of this thesis, there was no dataset publicly available for detecting the changes in streets. So the plan is to create a dataset containing pairs of images showing any change in the drivable area of the streets as the ‘change’ label or showing the same drivable area as the ‘no change’ label. The Oxford RobotCar Dataset [11] [12] is used to create the needed dataset. The utilized dataset contains the recorded data of manual driving in central Oxford, UK, from 6 May 2014 to 13 December 2015, including 1010.46 km recorded camera images, LIDAR point clouds, GPS coordinates, and INS. This dataset has few different trajectories that the car is driven for recording data. The most frequent traversed trajectory will be referred to as the main trajectory in this work, and it is shown in figure 3.1.

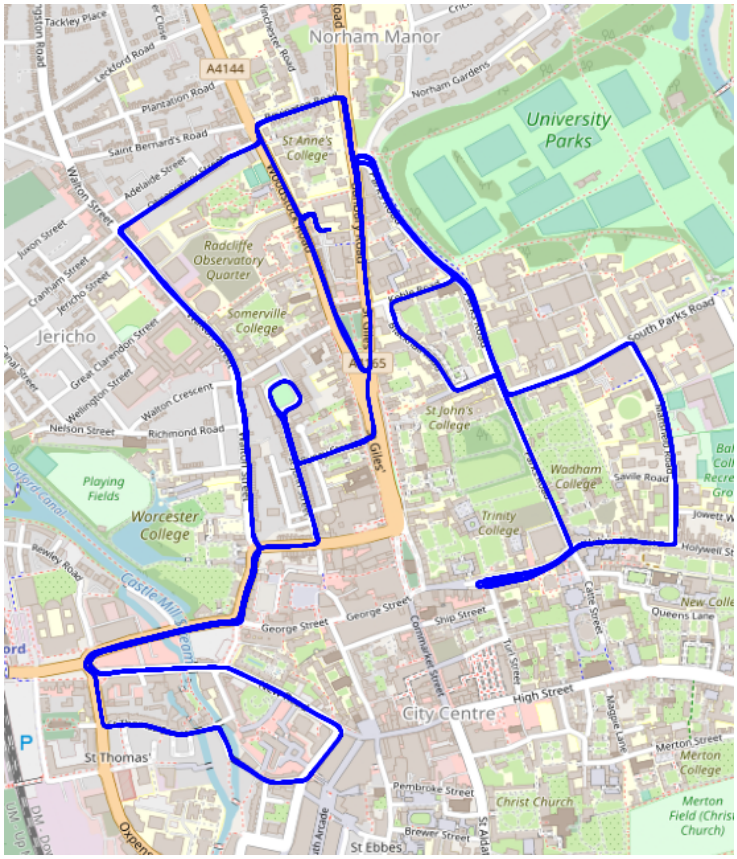



Figure 3.1. Main trajectory of Oxford Dataset

The main trajectory of the Oxford RobotCar Dataset is driven approximately 100 times during one year. Some roadworks have occurred within this period that affected the

drivable area. As discussed before, the dataset contains the images of streets taken by a camera mounted to the car. Moreover, these images are labeled with the geographical coordinates of the spot that the image is shot. Therefore, we can detect the exact location of changes in the streets' drivable area by comparing the corresponding camera images of the streets taken on different days. As a result, extracting the pairs of images showing the changes mentioned above will enable us to create a required dataset. Accordingly, two different datasets are created from the Oxford RobotCar Dataset, namely, the Change Detection Image Pairs (CDIP) dataset and the Large Change Detection Image Pairs (LCDIP) dataset.

## 3.2. CDIP Dataset

The CDIP is the first dataset that is created from the Oxford RobotCar dataset. To create the CDIP dataset, four different days of the Oxford RobotCar dataset are chosen, of which two are from the first days, and the other two are from the last days. As discussed before, each day of the Oxford dataset contains the geographical coordinates information of the trajectory and front view camera images showing the streets. Hence, to extract the changes in drivable area of the streets, camera images of the same spots can be compared together. Accordingly, The images of days '2014-11-18-13-20-12' and '2015-03-10-14-18-10' are compared to the images of days '2015-10-30-13-52-14' and '2015-11-13-10-28-08', respectively. This way, we have two trajectory pairs that are being compared to each other. Trajectories of each pair will be referred to as 'trajectory A' and 'trajectory B'. As stated above, the CDIP dataset is supposed to have pairs of images as instances of the dataset. In order to extract the pairs of images with the labels of 'change' or 'no change', the following steps are taken. Initially, images of each trajectory pair are compared manually. For each trajectory pair, there may be some parts of trajectory A and trajectory B that the same streets do not have the same drivable area, i.e, the drivable areas of those streets have been changed. These parts of trajectories with altered drivable areas of the road are annotated as 'change'. And some other parts of the trajectories A and B that have the same drivable area are annotated as 'no change'. These annotated sub trajectories will be referred to as 'segments'. After manually comparing the above-mentioned days, 18 segments are annotated as change and 14 segments as no change. The next step is to extract pairs of images from the segments. These image pairs are instances of a dataset, which shows the same or altered drivable area of a street. Therefore, they must have been taken from the exact same spot with almost the same viewpoint but on different days. Each image has its recorded time as a timestamp. Besides, the geographical coordinates of the trajectories, which are available by GPS and INS fused data, contain corresponding timestamps as well. By choosing the closest timestamp, the location of

each image can be obtained. Therefore, for each image of trajectory A, a corresponding image from trajectory B can be chosen by knowing the geographical coordinates of the images. The detailed process of extracting the image pairs is described as follows.

Each day in the Oxford RobotCar dataset contains the video track of the trajectory recorded by a camera mounted to the car. Each frame of the video has a timestamp specifying the recorded time, and there are location coordinates corresponding to each timestamp. So, the location of each video frame (image) can be determined using timestamps. As stated before, the pairs of images from the same locations can be extracted by finding the closest corresponding geographical coordinates of the images. These geographical coordinates are obtained by the timestamped fused GPS and INS data. However, even the locations obtained from the fusion of GPS and INS data can sometimes be inaccurate. This inaccuracy may hinder the performance of the above-mentioned approach for extracting the image pairs of the same spot and same viewpoint. To diminish the problem, the locational coordinates error between two trajectories can be calculated and compensated by considering the following conditions. Initially, the first images of each segment are selected manually, which means we can assure that the first images of two segments that are being compared together are from the same spot and viewpoint. Therefore, any difference between their corresponding location coordinates is an error. Secondly, assuming the GPS error usually remains the same within a limited time and distance, the GPS error will stay almost the same for the current segment. Finally, nearly all the streets that the Oxford RobotCar Dataset's car is driven through have only one lane for each direction. And about the streets with multiple lanes, the car is always driven in the left-most lane, so the trajectories of the car on different days align on the same lanes. Considering these three conditions, the GPS error between the trajectory pair of a segment can be calculated by subtracting the location coordinates of trajectory A's first image from the coordinates of trajectory B's first Image. The result will be a two-dimensional vector representing the relative geographical coordinates error between two trajectories. This error can be compensated by adding the obtained error vector to all the images' coordinates of trajectory A. As a result, the coordinate error between the same spots of two trajectories of a segment is minimized. So far, each compared trajectory pairs of a segment have images of the trajectory while the error of location coordinates is minimized using the above-mentioned approach. As the last step of extracting image pairs, for all images in trajectory A and B, pairs of images are extracted such that for each pair, the first image belongs to trajectory A of the segment, the second image belongs to the trajectory B of the segment, and the images of the pair instance have the closest location coordinates among other images. The label of the image pair is specified based on the label of the segment from which this image pair is extracted.

Concisely, the CDIP dataset is created by comparing the recorded images of two trajectory pairs. Eighteen segments are annotated as ‘change’ and fourteen segments as ‘no change’. After extracting the annotated image pairs, the final dataset consists of 11429 image pair instances. 49.76 percent of the instances have positive labels i.e. the image A and image B of an image pair instance are showing a change in the drivable area, and 50.24 percent of the instances have negative labels i.e. the image A and image B are showing no change in the drivable area. Some examples of positive pairs and negative pairs are shown in figure 3.2.


Figure 3.2. CDIP Dataset Image Pair Samples

## 3.3. LCDIP Dataset

As discussed in section 3.2, the CDIP dataset is created by comparing the trajectories' images that exist in the Oxford RobotCar dataset. The Large Change Detection Image Pair (LCDIP) dataset is created to increase the number and diversity of the instances compared to instances in the CDIP dataset. In generating the CDIP dataset, four days of the Oxford dataset were chosen, and as a result, there were two trajectory pairs to extract the image pairs. The LCDIP dataset, however, is generated by using the trajectories of ten days. Moreover, the trajectory images of each day in the LCDIP dataset are compared to all other days. As a result, not only the number of instances will be larger than in the CDIP dataset, but the diversity of the instances will increase as well. This increment in diversity of the LCDIP dataset instances happens because each image from each segment of the trajectory is compared to the same segment of all other days. Whereas, in the CDIP dataset, each image from each segment is compared to the same segment of only one day. Accordingly, the LCDIP dataset contains multiple instances from the same spot with different lighting, weather conditions, and surrounding objects like cars and pedestrians. The second significant difference between two datasets is related to the segments. In the CDIP dataset, the segments are annotated as 'change' or 'no change'. In the LCDIP dataset, however, segments do not have any annotations. Segments of a day in the LCDIP dataset are manually selected as parts of the trajectory that the drivable area is different compared to all or more of the other days. Then, for each manually extracted segment of a day, the corresponding part of all other days' trajectories are extracted manually and referred to as sub-trajectories. This selection is made manually such that the viewpoint and location of the first and last images of each segment match the viewpoint and location of the first and last images of all the corresponding sub-trajectories respectively. An example of a segment and its corresponding sub-trajectories is provided in figure 3.4. Accordingly, each of the ten days has some segments, and each segment has its corresponding sub-trajectories from other days that the start and endpoints are chosen manually. The selected segments of ten days are shown in figure 3.3.

The next step is to extract the annotated image pairs to create the LCDIP dataset. To generate instances with 'change' labels, image pairs are extracted from each segment and its corresponding sub-trajectories from other days. And to generate the instances with 'no change' labels, image pairs are extracted by selecting two corresponding sub-trajectories of a segment. The process is shown in figure 3.4. The approach for extracting the images from video frames is the same as described in section 3.2.

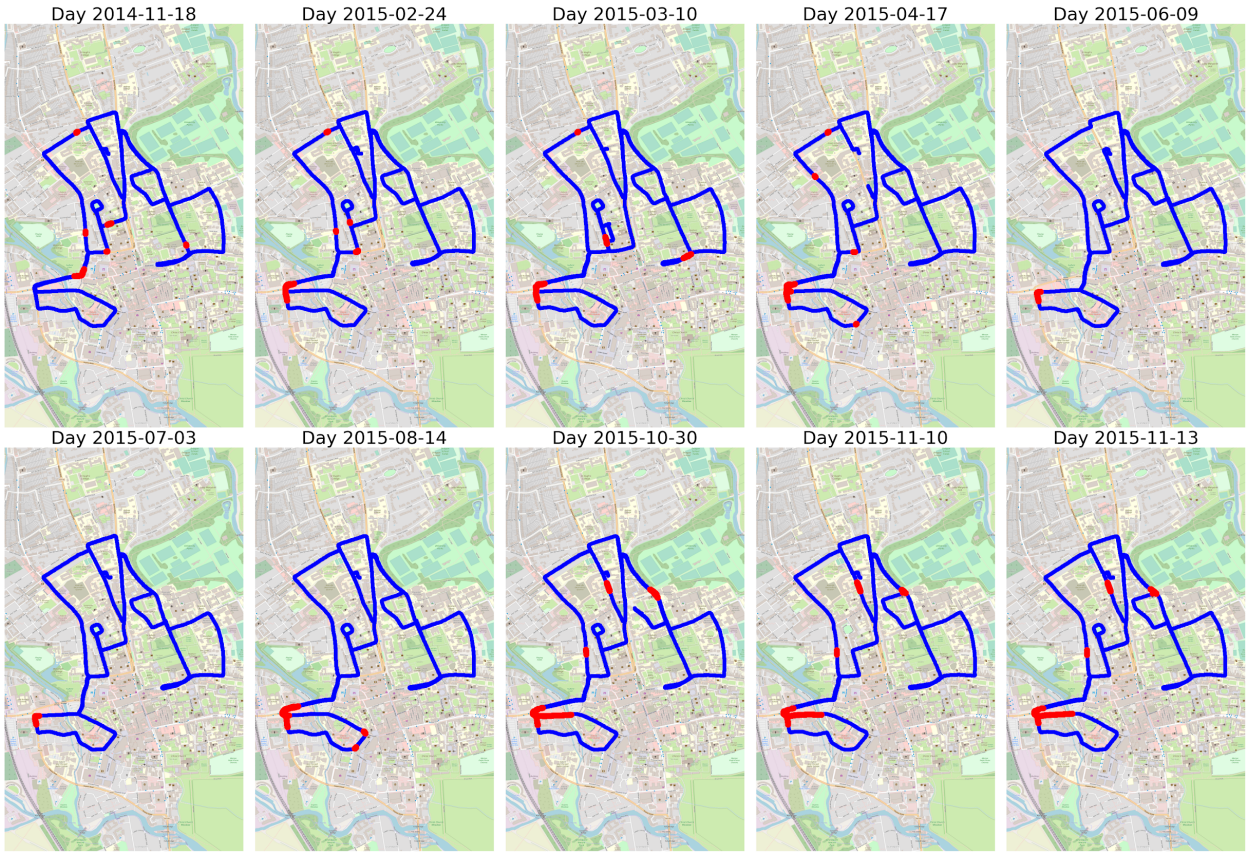



Figure 3.3. Extracted segments of 10 days for the LCDIP dataset. The red color illustrates the selected segments.

Finally, all segments are grouped into root-segments with an id based on their location on the trajectory. Root-segments are illustrated in figure 3.5. To conclude, the LCDIP dataset is created by using the images of trajectories of ten days. Sixty-two segments are selected from ten days and grouped into 16 root-segments. After extracting the image pairs, the final dataset consists of 177889 image pair instances. 21.92 percent of the instances have positive labels i.e. the image one and image two are showing a change in the drivable area. 78.08 percent of the instances have negative labels, i.e., image one and image two show no change in the drivable area. Subsequently, the LCDIP dataset has a diverse and larger number of instances. Unlike the CDIP dataset that has separate ‘change’ and ‘no change’ segments from different parts of the trajectory, positive and negative instances of the LCDIP dataset are from the same segment, i.e., an image from a segment appears both in a positive instance and a negative instance. An example of extracting positive pairs and negative pairs is shown in figure 3.6.

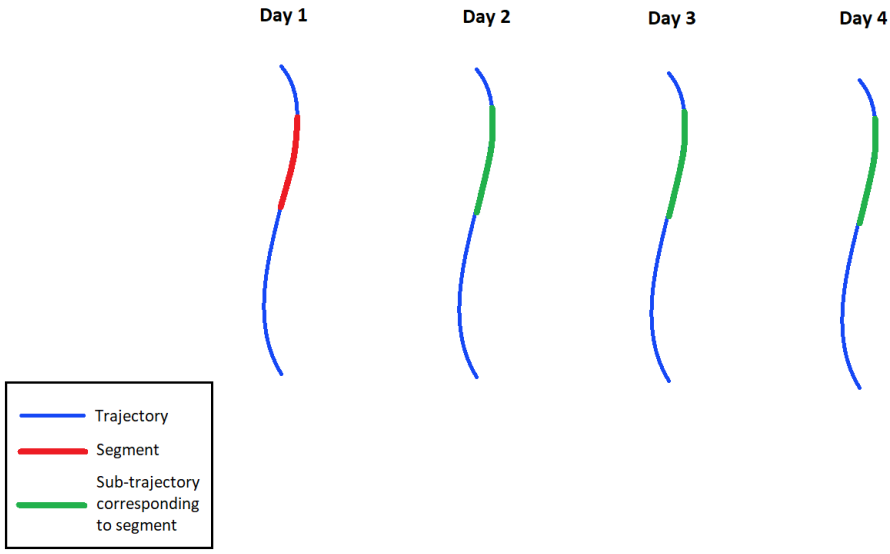



Figure 3.4. Process of extracting image pairs from segments. The red color is indicating the detected segment from day one. The green color indicates the corresponding sub-trajectory to the detected segment, with the same start and endpoint. Positive image pairs are extracted by considering day one's trajectory as trajectory A and each other days' sub-trajectories as trajectory B. Negative image pairs are extracted by considering all combinations of sub-trajectories of days 2, 3, and 4 as trajectories A and B regardless of the orders.

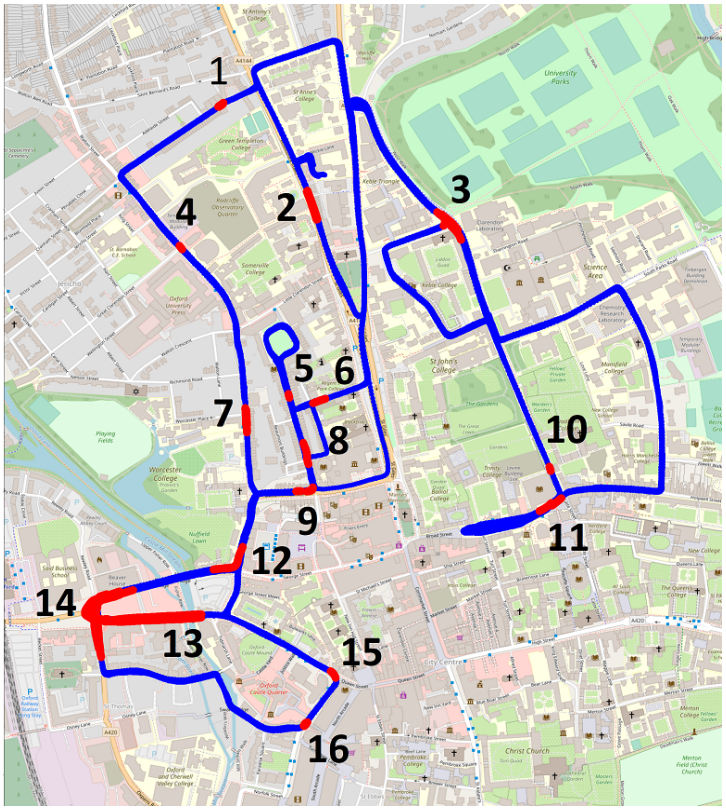



Figure 3.5. All segments are grouped into root-segments.


Figure 3.6. LCDIP Dataset Image Pair Samples

## 3.4. Train And Validation Split

The details of the generated CDIP and LCDIP datasets are presented in sections 3.2 and 3.3. This section discusses the process of splitting the datasets to train and validation sets for training the model and evaluating the generalization capacity of the model. Moreover, section 3.4.3 explains creating validation folds for cross-validation experiments.

### 3.4.1. CDIP Dataset Split

According to the dataset chapter, two datasets are created for this work using the Oxford RobotCar Dataset, namely, Change Detection Image Pair (CDIP) and Large Change Detection Image Pair (LCDIP) datasets. As stated before, the CDIP dataset contains 11429 image pair instances. Each instance consists of two images showing the same spot of a street on four different days. The drivable area of the road visible in the images of an instance can be the same or different, i.e., the path that a car can go on the road visible in the image could have been changed due to a road construction process. The image pair instances are labeled as ‘change’ or ‘no change’ accordingly. Roughly half of all instances (49.76%) have positive (change) labels, and 50.24% of the instances have negative (no change) labels. To use the dataset for training an artificial neural network model, it has to be divided into the train, validation, and test sets. This split can be done randomly sometimes. In the CDIP dataset, however, some instances are similar to each other because they show the same street. In other words, the CDIP dataset instances are images of the streets, and a specific street segment is covered by multiple image pair instances. And, many dataset instances show the same or an altered drivable area of a particular road segment. So, by splitting the instances to train/validation/test set randomly, image pair instances of a single street segment may appear in all train, validation, and test sets simultaneously. This is not acceptable because the data in the validation or test set must not appear in the train set, otherwise, this occurrence will lead to a higher but invalid artificial neural network model performance on the validation and test sets. Therefore, to prevent such high and invalid performance measures, we cannot divide instances of the CDIP dataset to train, validation, and test sets randomly. This division, therefore, is performed based on the road segments manually such that all the image pair instances corresponding to a particular road segment appear only in one of the divided sets. At the same time, there are not many road segments available in the dataset. As discussed in section 3.2, the CDIP dataset contains 18 positively labelled road segments and 14 negatively labelled road segments. Thus, there are not enough positive and negative road segments to

have large and diverse enough train, validation, and test sets. So, we decided to have only train and validation sets. This way, we increased the training and validation set's diversity and size by losing the chance to further assess generalisation capacity of the model on the test set. In conclusion, the CDIP dataset is divided into train and validation sets. The train set contains instances from thirteen positively labelled road segments and ten negatively labelled road segments resulting in around 10000 image pair instances. The validation set includes instances from five positively labelled road segments and four negatively labelled road segments resulting in about 1000 image pair instances. Train and validation split is illustrated in figure 3.7.

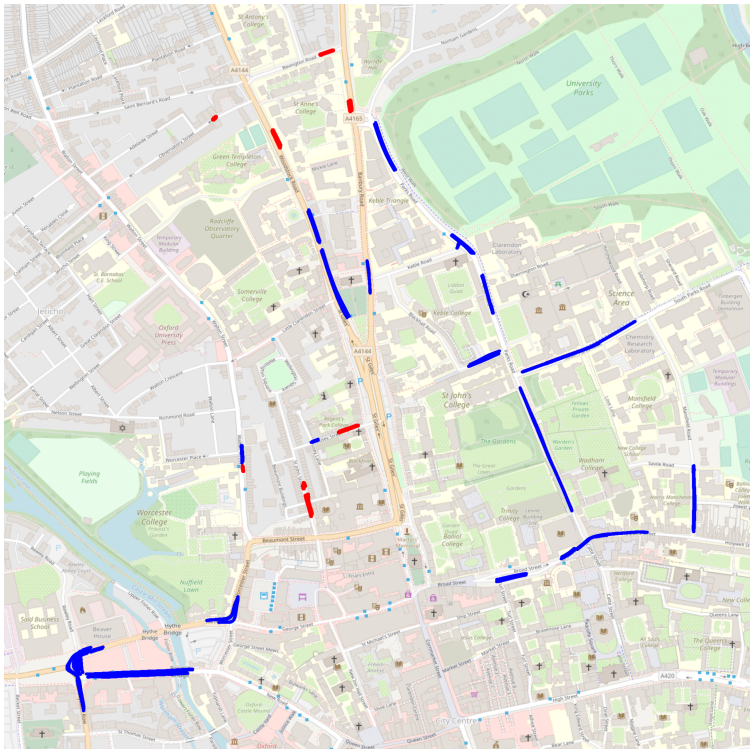


Figure 3.7. Train and validation split of the CDIP dataset. The blue path image instances are used as train instances. And the red path image instances are used as validation instances.

### 3.4.2. LCDIP Dataset Split

In a similar manner, to use the LCDIP dataset for training of the artificial neural network, it has to be divided into the train and validation sets. The test set is discarded because of the lack of enough data like in the CDIP dataset split. The LCDIP dataset contains 177889 image pair instances, of which 21.92% have positive labels, and 78.08% have negative labels. Images of a positive dataset instance show a change in the drivable area of a road, and images of a negative instance show the same drivable area. As stated in the dataset chapter, the image pair instances of the LCDIP dataset are extracted from 16 road sections (root-segments). Therefore, like the CDIP dataset, there are not enough road sections to divide the LCDIP dataset into the train, validation, and test sets. Thus, similar to the CDIP dataset, instances of the LCDIP dataset are split into train and validation sets based on the root-segment id manually. Accordingly, the validation set contains the image pair instances extracted from the root-segments 2, 3, 5, and 15, resulting in 31706 instances. The rest of the instances are gathered into the train set, resulting in 146183 image pair instances. Train and validation split is illustrated in figure 3.8. The CDIP dataset has an almost balanced number of positive and negative (‘change’ and ‘no change’) instances. Thus its train and validation sets are also balanced. The LCDIP dataset, however, has an imbalanced number of positive and negative instances. Therefore, the train and validation sets of the LCDIP dataset are also imbalanced. Most of the machine learning algorithms expect a balanced dataset [13]. Otherwise, the classification model trained on the imbalance data will have a good performance classifying the instances of the majority class, while the minority class instances will be misclassified frequently. Therefore, the train set of the LCDIP dataset must have a balanced number of positive and negative examples to prevent biased training. Accordingly, to solve the problem of having imbalanced train and validation sets in the LCDIP dataset, only a limited number of the instances that have the majority class will be selected for each training or validation epoch. This approach is known as undersampling. Undersampling will happen randomly such that the number of the chosen majority class instances is approximately the same as the number of available minority class instances in the set. Using this approach, each time the train set is used in a training epoch or a validation set is used to measure the model’s performance, the chosen instances for training or validation have an approximately balanced number of both positive and negative class instances.

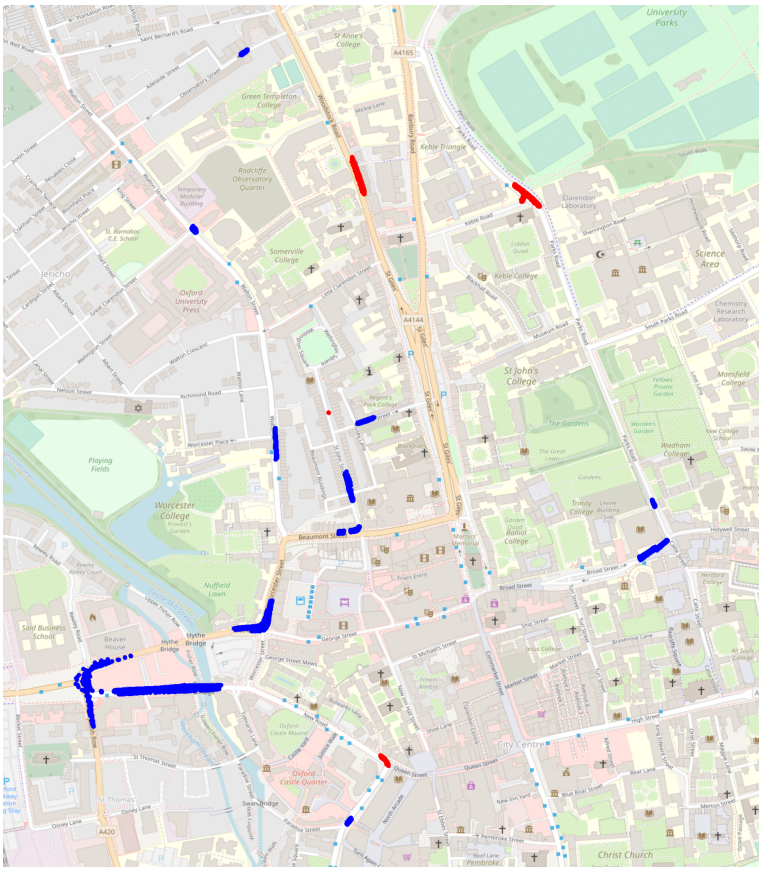



Figure 3.8. Train and validation split of the LCDIP dataset. The blue path image instances are used as train instances. And the red path image instances are used as validation instances.

## 3.4.3. LCDIP Dataset Cross-Validation

In addition to evaluating the artificial neural network model with a single validation set, it is beneficial to assess the performance on more than one individual train/validation splits. This way, we can see the model's performance on the validation sets with different divisions and find the average performance instead of a single assessment on a single split. Evaluating the model on only one train/validation split could result in a lucky (i.e. affected by chance) high performance or unlucky low performance than the average. Thus, cross-validation experiments are conducted to find the average performance of the artificial neural network on the LCDIP dataset. To create the validation folds following steps are followed. The LCDIP dataset consists of 16 root-segments. These root-segments are randomly permuted for N times to create N different random permutations of the root-segments. Then, from each random permutation, M subsets of root-segments are selected to form M validation folds. This selection is performed such that for the root-segments random permutation $N_i$ , the root-segments are chosen for the validation fold $M^{Ni}_1$ respectively until the fold $M^{Ni}_1$ has at least 400 positive instances and 400 negative instances. Then, the root-segments will be selected for the next validation fold according to the order of permutation $N_i$ until the fold reaches the mentioned minimum limits. This process will continue till there is no root-segment remaining in the $N_i$ permutation. The number of folds from each root-segments permutation may vary. The total number of validation folds is calculated by $\sum_{i=1}^{N} M^{Ni}$ . Here, $M^{Ni}$ is the number of folds extracted from the root-segments permutation $N_i$. For each validation fold, the instances of the root-segments that exist in the validation fold will be used as the validation set, and the rest of the instances will be used as the train set.

# 4. Methods

This chapter describes the methods that are used in this work. In section 4.1, the proposed pipeline and an overview of the utilized neural networks are provided. Section 4.2 describes the Principal Component Analysis (PCA) technique. Furthermore, the score-weighted visual explanations for convolutional neural networks (Score-CAM) [14], [15] method is discussed in section 4.2.

## 4.1. Proposed Pipeline

Artificial Neural Network is a method used in machine learning, inspired by the neurons in the human brain [16]. According to [16], an artificial neural network is a set of interconnected nodes known as artificial neurons, and it has three major components. The first component is the details about the neurons, such as the number of inputs, number of outputs, the weights, and the activation function, which is a function to calculate the output value based on the input values. Second is the topology of the network, which determines the neurons' architecture and the way they connect. The last component is the learning rules. These learning rules specify how the weights of the connections among neurons have to be initialized and how they should be updated [16]. The artificial neural networks (ANN) have been extensively used for solving classification problems [17]. As stated in [17], the ANNs have successfully solved different practical classification problems in business and industries. Accordingly, the artificial neural network method is chosen to solve the drivable area change detection problem. Thus, an ANN model is proposed in this work. The proposed model contains two components. The first component takes in two images as input and produces the corresponding feature-maps of the input images. These inputs are images of a particular road section with its surrounding environment taken by a camera mounted on a car, as discussed in the dataset chapter. The extracted feature-maps, then, are aggregated and fed into the second component. The second component is responsible for comparing the extracted feature-maps and outputting the classification label, i.e., it has to determine whether the extracted feature-maps of the input images imply a change in the drivable area or not. The proposed pipeline is illustrated in figure 4.1. To perform the feature-map extraction task, the Convolutional Neural Network (CNN) is used in the former component, and Multilayer Perceptrons (MLP) are used in the latter component to create the final classification labels. More details about the CNN and MLP are presented in sections 4.1.1 and 4.1.2, respectively, as follows.

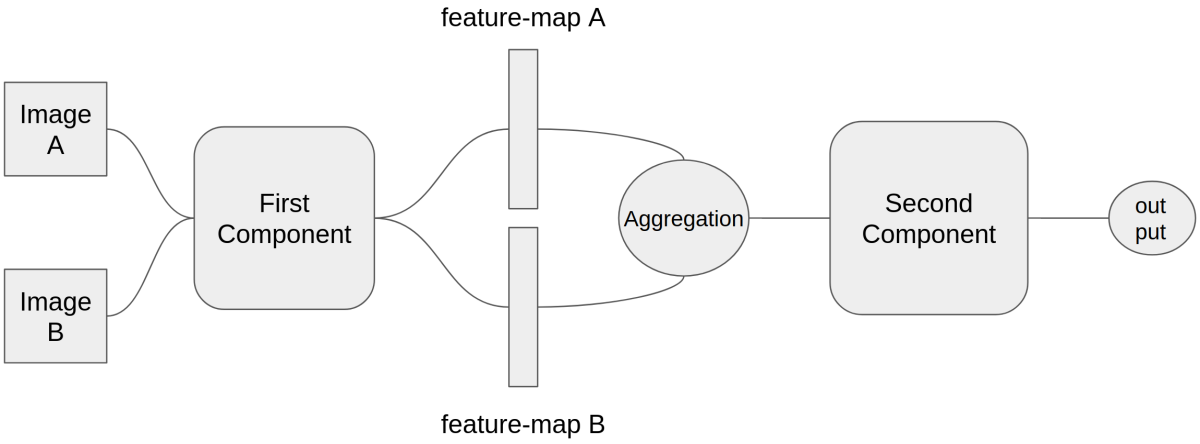



Figure 4.1 Proposed pipeline for change detection problem

## 4.1.1. Convolutional Neural Networks (CNN)

As discussed in section 4.1, The first component of the proposed ANN model takes in two images as input and extracts the corresponding feature-maps. The Convolutional Neural Network structure is used to extract the feature-maps. The Convolutional Neural Network (CNN), inspired by the human visual cortex, is a deep artificial neural network. In convolutional neural networks, the neurons are responsible for extracting the features from the input images [18]. Thus, the convolutional neural networks have been used to extract the features of input images in computer vision tasks like object detection, image recognition, etc [19]. There are many pre-trained convolutional neural network models publicly available such as ResNet [20], VGG [21], GoogleNet [22]. Some of these architectures can surpass the human-level accuracy on object recognition tasks [23]. To take advantage of pre-trained feature extraction networks' performance, 50-layer ResNet [20] is selected to be used as the first component of the proposed neural network architecture to extract the feature-maps of the input images.

ResNet-50 is a fifty-layer deep convolutional neural network [20]. The architecture of all ResNet models is illustrated in figure 4.2. According to [20], the input of the ResNet is an image of size 224 x 224 with three channels for RGB values. ResNet is a residual neural network. Unlike plain neural networks, residual neural networks have shortcuts among the layers by skipping some layers. The motivation of jumping over the layers is to overcome the problem of vanishing gradients [24]. The artificial neural networks use

gradient-based learning methods to update the weights by backpropagating the gradient of the loss function. The problem is that backpropagated gradients are getting smaller as they move away from the output layer towards the input layer [25], [26]. This problem is known as vanishing gradients, which hinders the process of updating the weights. This may completely stop the deep neural networks from further training in the worst case. According to the architecture of ResNet-50, the output of the fifth conv-block is a flattened feature-map of size 2048, which is fed to an average pooling layer followed by a fully connected layer of size 1000 with softmax activation as output layer. ResNet-50 has been trained on more than a million images from the ImageNet dataset [27]. As mentioned before, the first component's main goal is to extract the feature-maps of the input images, so the Resnet-50 with pre-trained weights is used for this task. However, the utilized Resnet-50 is excluding the last fully-connected layer with softmax activation so that the output of the ResNet-50 becomes a feature-map vector of size 2048.

| layer name | output size | 18-layer | 34-layer | 50-layer | 101-layer | 152-layer |
|---|---|---|---|---|---|---|
| conv1 | 112×112 | 7×7, 64, stride 2 | | | | |
| conv2_x | 56×56 | 3×3 max pool, stride 2 | | | | |
| | | $\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$ | $\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$ | $\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$ | $\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$ | $\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$ |
| conv3_x | 28×28 | $\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$ | $\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$ | $\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$ | $\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$ | $\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$ |
| conv4_x | 14×14 | $\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$ | $\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$ | $\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$ | $\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$ | $\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$ |
| conv5_x | 7×7 | $\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$ | $\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$ | $\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$ | $\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$ | $\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$ |
| | 1×1 | average pool, 1000-d fc, softmax | | | | |
| FLOPs | | $1.8\times10^9$ | $3.6\times10^9$ | $3.8\times10^9$ | $7.6\times10^9$ | $11.3\times10^9$ |

Figure 4.2. ResNet architecture [20].

### 4.1.2. Multilayer Perceptron (MLP)

After extracting the feature-maps of the input images by the first component of the proposed model, the feature-maps have to be compared and classified as ‘change’ or ‘no change’ in the second component. As discussed before, artificial neural networks have demonstrated remarkable performance in classification tasks. Therefore, to solve the change detection problem, the multilayer perceptron network structure is used in the second component of the proposed model to perform the feature-maps comparison task. The multilayer perceptron is a feed-forward neural network consisting of an input layer, one or more hidden layers, and an output layer [28]. The input layer’s size is the same as the number of features that the input data has. The output layer and each hidden layer consist of multiple artificial neurons. The size and number of hidden layers is a variable parameter of the network architecture, but the output layer has the same size as the number of classes of the classification problem. The connections’ flow is always from the input layer towards the output layer, i.e., each layer’s input flow is supplied by the previous layer closer to the input layer, and the output flow is connected to the next layer closer to the output layer. The neurons of a layer are not interconnected [28]. The architecture diagram of multilayer perceptron networks is provided in figure 4.3.

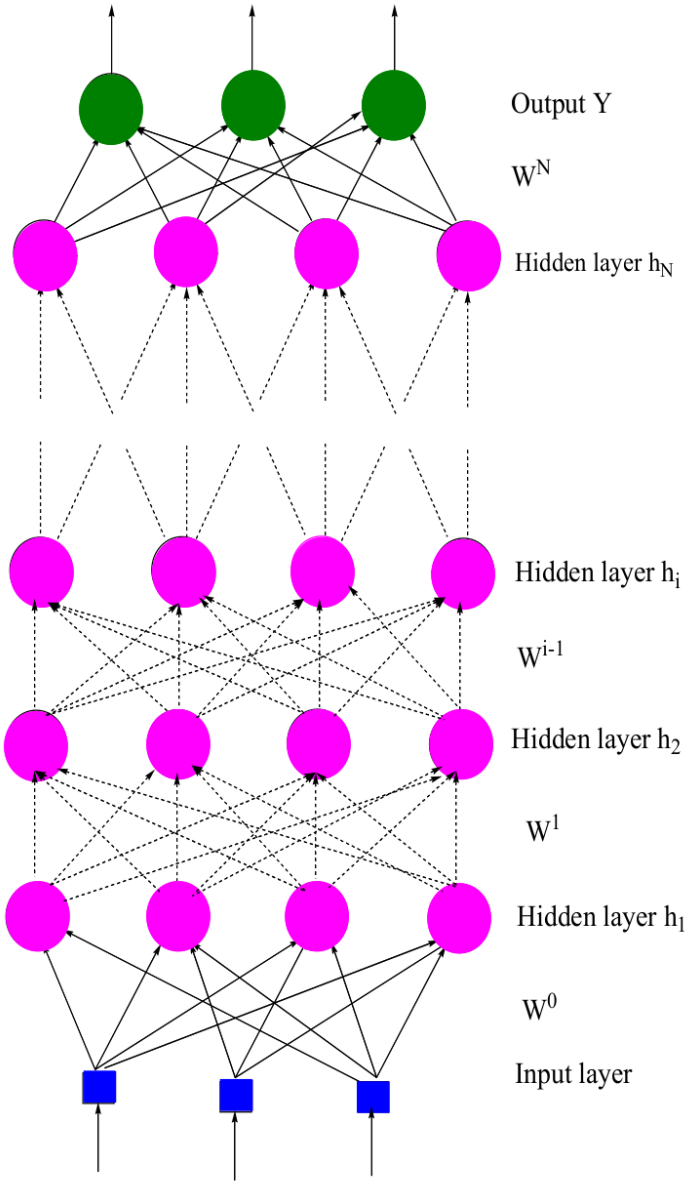



Figure 4.3. Multilayer perceptron structure [28]

## 4.2. Principal Component Analysis (PCA)

Principal Component Analysis is a mathematical technique used for reducing the dimensionality of data features without losing much information [29]. The PCA method creates new variables instead of the original variables of the data that are called Principal Components. These new variables are linear combinations of the data's original features. As stated in the article [29], the principal components are variables in directions aligned with the variation of original values. In other words, the first principal component (PC1) is a variable in the direction in which the original values have the maximum variation. The second principal component (PC2) is a variable in the uncorrelated direction to the PC1 in which the original values have the largest variation. Accordingly, the principal components are ordered based on the variation in the original features that each principal component represents. The steps of calculating the principal components are as follows. First, each original feature of the data must be centered to zero average. The next step is to calculate the covariance matrix of the zero-centered original features. After calculating the covariance matrix, the principal components are the normalized eigenvectors of the covariance matrix ordered by the variation level.

As the primary goal of the proposed change detection model's first component is to extract the input images' feature-maps, it will be beneficial to evaluate the first component's performance before building the next component. One way of grasping information from data is to visualize it. A two-dimensional diagram can visualize two variables, and a three-dimensional diagram can visualize three variables. In this problem, however, the first component's output feature-map is a vector of size 2048. Therefore, the dimension of the feature-map has to be reduced to two so that a 2D plot can visualize it. As discussed above, PCA is a technique to reduce the dimensionality of the data. Hence, the PCA method is used to reduce the dimension of the first component's output feature-map to a two-dimensional vector.

## 4.3. Score-CAM

As discussed in section 4.1.1, the proposed artificial neural network for the change detection problem consists of the image feature extraction component and the change detection component. As stated in section 4.2, the PCA method is used to evaluate the proposed model's feature extraction component. Accordingly, by plotting two principal components of the extracted features, we can gain some intuition about the extracted features. Using this technique, however, it is not possible to understand why or how the model has a good or bad performance in extracting the feature-maps from input images.

Besides, the artificial neural networks are called mysterious black boxes because there is no comprehensive understanding of the artificial neural networks' internal organization or their optimization process [30], [31]. Many methods try to explain convolutional neural networks in particular, such as Gradient visualization [32], Perturbation [33], and Class Activation Map (CAM) [34]. The proposed method in the paper [14], score-CAM, is one of the Class Activation Map (CAM) visual explanation methods to illustrate the convolutional neural networks' hidden organization. This method provides a visual explanation by an attribution map highlighting the input image's critical regions, which affect the convolutional neural network's final output. Therefore, the score-CAM method is used for further interpretation and evaluation of the proposed change detection model's first component. A more detailed explanation of the score-CAM method is provided as follows.

Convolutional neural networks usually contain multiple convolutional layers with different channels and activation functions [35]. Each channel with corresponding kernel and activation function in each layer will lead to a 2D vector. This vector is called the activation-map of the corresponding channel of the layer [35]. According to article [14], score-CAM method provides a visual explanation that illustrates the important regions of the input image by a weighted combination of the last convolution layer's activation-maps. Let us consider N as the number of channels in the last convolutional layer. Accordingly, for a single input image sample, N activation-maps are extracted from the last convolutional layer of the network. Extracting these feature-maps is considered phase one. Activation-maps then are upsampled to be the same size as the input image. Each upsampled activation-map acts as a mask on the original input image. In phase 2, the N masked images are fed into the convolutional neural network followed by fully connected layers. The result is the N number of outputs from the model. The final score-CAM is a weighted combination of the obtained activation-maps in phase one, in which the weights are the outputs of the model in phase two. In classification problems, the output is usually a vector of the same size as the number of classes [17]. Therefore, in classification problems, the score-CAM of different classes can be extracted by selecting the corresponding class probability as the weight for the score-CAM generation [14]. The process is illustrated in figure 4.4.

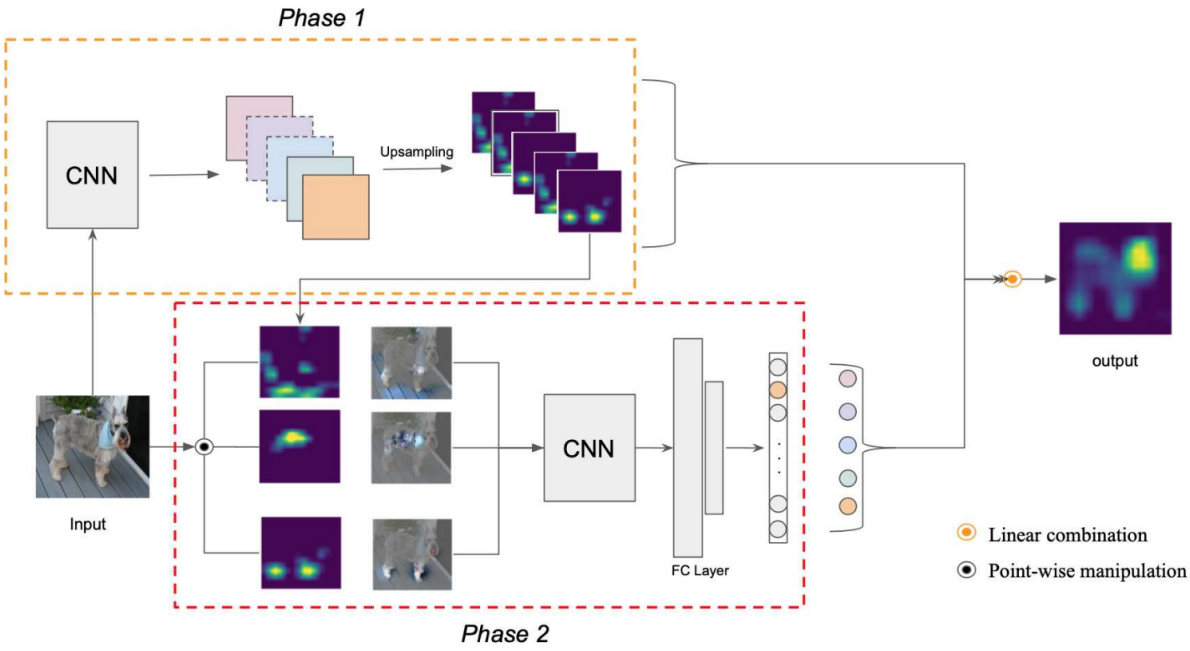



Figure 4.4. The pipeline of the Score-CAM method. The CNN module is the same in the phase one and two [14].

# 5. Experiments

This work focuses on detecting the changes in the drivable area of the roads. The artificial neural network method is utilized to classify images from the same road as if they contain a change or not. The general pipeline of the proposed artificial neural network is provided in the methods chapter. Considering the provided pipeline, the optimal network architecture and hyperparameters have to be found for solving this problem. Such that the neural network predicts the majority of data correctly with as few false predictions as possible. To find such optimal network structure and hyperparameters, a few tests and experiments are designed and carried out. Details of the experiments are provided in chronological order.

## 5.1. Pre-trained ResNet as Feature-Map Extractor

As the first experiment for finding the optimal network structure and hyperparameters, the pre-trained ResNet-50 model is used. The pre-trained ResNet-50 is a feature-map extractor convolutional neural network model that is trained on the ImageNet dataset. The pre-trained ResNet-50 model is used for extracting the feature-maps from the input images as the first component of the proposed pipeline. Input images are pair images of a road section that might show the same or different drivable area. For instance, if the drivable area of a street is altered due to road work, the image pair will show a different drivable area of the street. Each pair image contains two images that will be referred to as image A and image B. The pre-trained ResNet-50 model extracts feature-map A and feature-map B from image A and image B. Each extracted feature-map is a vector of size 2048. These two feature-maps have to be unified so that the unified feature-map can be fed into the second component. The feature-maps are unified by using two different approaches. Multiple unifying approaches are used to analyze the effect of each approach on the final performance of the neural network model. One approach is to concatenate the feature-maps A and B resulting in a vector of size 4096. The model that uses this approach is named the ‘unify by concatenation’ model. The other approach is to calculate the subtraction and take the absolute value using the following formula| $featuremap\ A\ -\ featuremap\ B$ |. The second approach results in a vector of size 2048. The model that uses this approach is named the ‘unify by absolute subtraction’ model. The unified feature-map vector is then fed into the second component as an input to be classified as if the drivable area of the road is different or not.

The second component comprises an input layer, a single dense layer of size 100 neurons with the ReLU activation function and L2 regularization, and an output layer. Also, there are two dropout layers: one is after the input layer, and the other is after the dense layer. The output layer of the second component has two neurons with the softmax activation function to predict the probability of the final ‘change’ or ‘no change’ classes. The Adam optimizer with a learning rate of 1e-6 is used in this experiment. And the CDIP dataset is used to train the model. The architecture of the second component is briefed in table 5.1. And the architecture of the full model of this experiment is displayed in figure 5.1.

| Layers | Size | Activation | L2 Regularization | Dropout Rate |
|---|---|---|---|---|
| Input | 4096 / 2048 | | | |
| Dropout | | | | 0.1 |
| Dense | 100 | ReLU | 0.01 | |
| DropOut | | | | 0.1 |
| Output | 2 | Softmax | | |

Table 5.1. Network architecture and hyperparameters of the second component. The input size of the second component of the ‘unify by concatenation’ model is 4096. The second component of the ‘unify by absolute subtraction’ model has an input size of 2048.

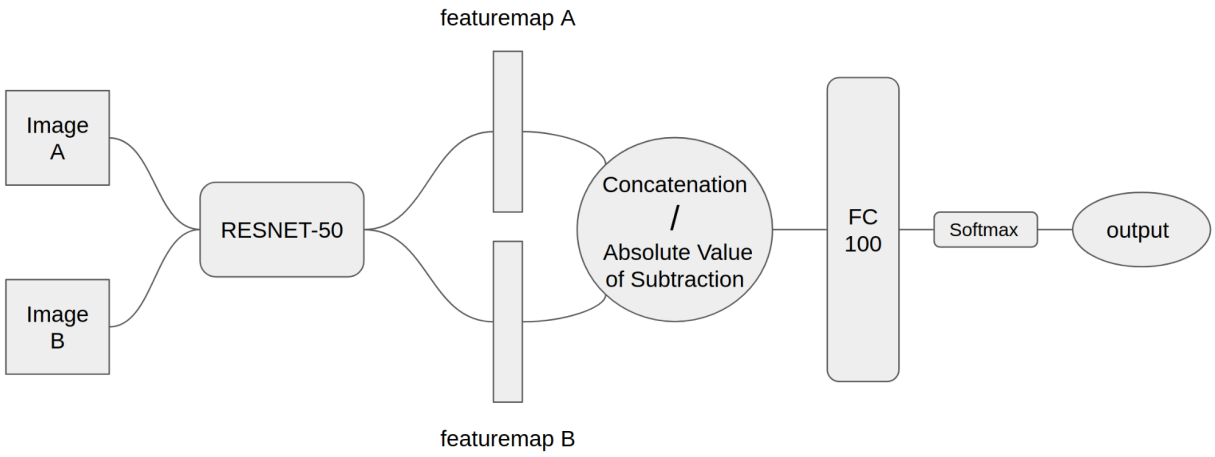



Figure 5.1. Full Model Architecture of the Experiment

Both components are trained as a single neural network model for 20 epochs. The training process is stopped after 20 epochs because the validation losses are not decreasing and the validation accuracies are not increasing. The training history is illustrated in figure 5.2. This plot shows extreme overfitting that is happening from the first epoch such that the validation accuracies do not increase at all.

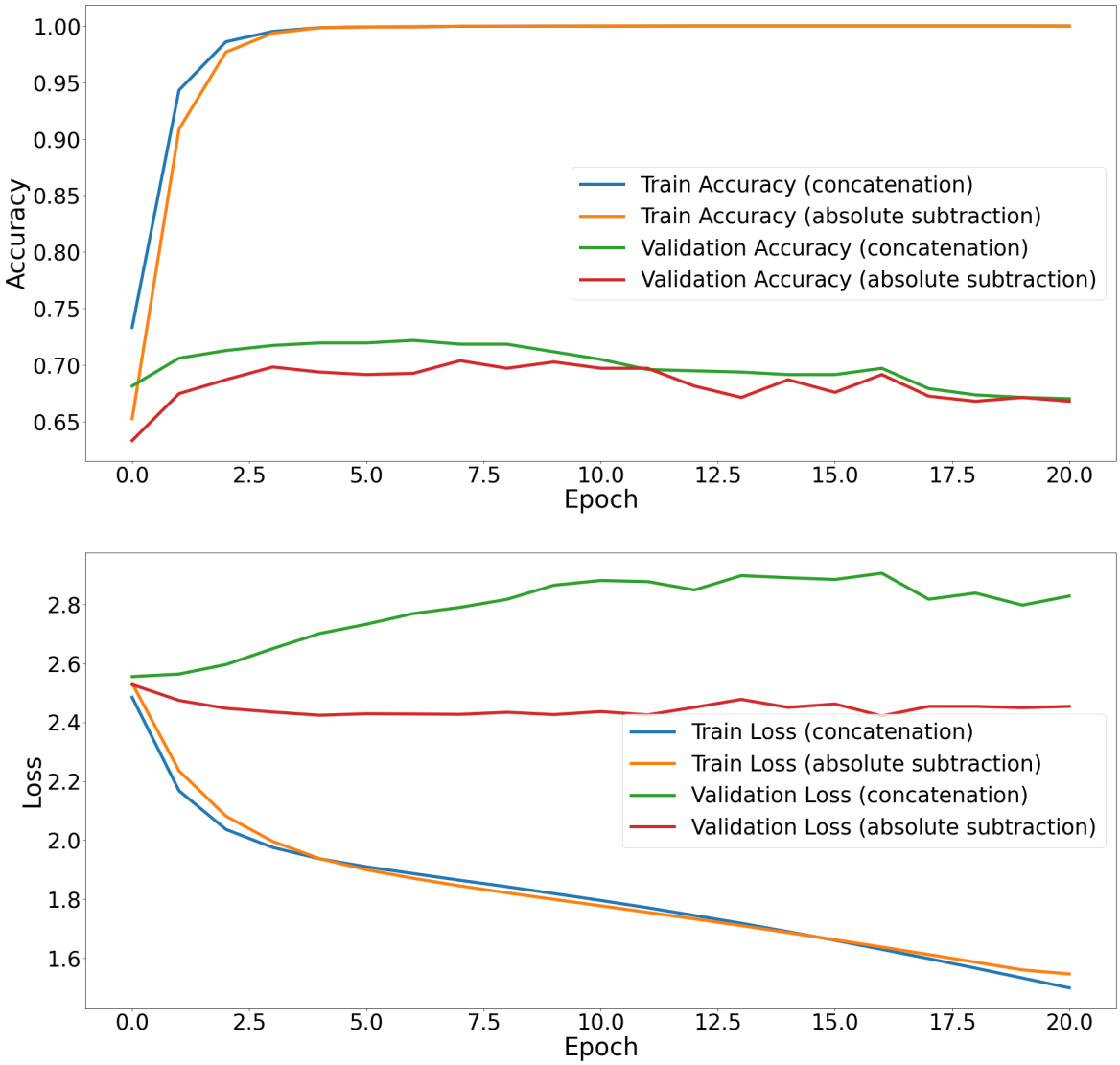



Figure 5.2. Training history of the two proposed neural network models. One of the models is concatenating the extracted feature-maps by the first component. The other model calculates the absolute value of the subtracted feature-maps.

Training the whole neural network model (both components) does not seem to improve the performance of the model as it is visualized in figure 5.2. The first component is a pre-trained ResNet-50 model that is trained to extract the feature-maps from the images. So, the ResNet-50 model with pre-trained weights can be used as the first component without further training. Since training both first and second components did not show any progress, the weights of the pre-trained ResNet-50 model are made frozen during the training process to analyze its effect on the final performance of the model. After carrying out few experiments, we noticed that by extracting the feature-maps using the first component separately and training the second component afterward, the training process is much faster. Also, the final performance of the model is relatively higher this way. As the weights of the first component are frozen, the first component's output for each input will always be the same during the training process. Therefore, to speed up the training process, we decided to extract the feature-maps of all images using the ResNet-50 model and store them separately. Then, instead of extracting feature-maps during the training process, the corresponding feature-maps are loaded, unified, and fed into the second component as an input.

The second component of the model is trained for 300 epochs. The training process is stopped after 300 epochs because the validation losses stopped decreasing. Also, the validation accuracies did not improve after 100 epochs. The training history plots are shown in figure 5.3. The training history plots illustrate that the gap between the training and the validation accuracy starts from the early epochs and grows larger as the number of training epochs increases. These training history plots show considerable overfitting of both 'unify by concatenation', and 'unify by absolute subtraction' models on the train set. From figure 5.3, it is visible that the 'unify by concatenation' model performs slightly better than the 'unify by absolute subtraction' model. This means concatenating the feature-maps is a better way to unify the feature-maps in this experiment. Few examples of the true and false prediction of the 'unify by concatenation' and 'unify by absolute subtraction' models on the validation set are presented in figures 5.4 and 5.5. The best performance of each model is presented in table 5.2.

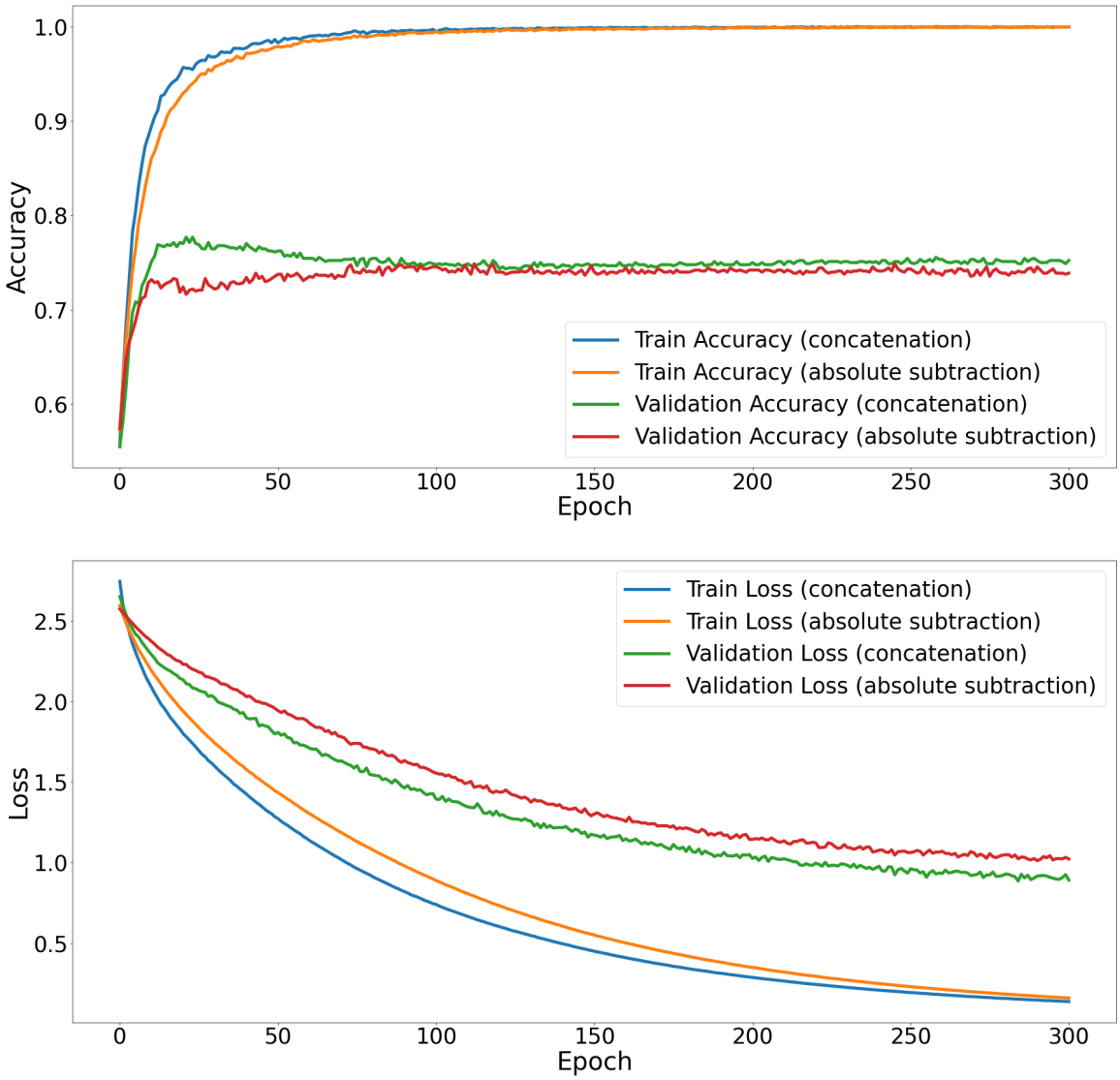



Figure 5.3. Training history of the two proposed neural network models. The first component is not included in the training process. Instead, the feature-maps of the input images are extracted and stored separately. Corresponding feature-maps are then loaded, unified, and fed into the second component during the training process. One of the models is concatenating the extracted feature-maps. The other model calculates the absolute value of the subtracted feature-maps to unify them.


Figure 5.4. Prediction samples of the ‘unify by concatenation’ model on the validation set. Rows are illustrating true-positive, false-positive, true-negative, and false-negative prediction instances respectively.


Figure 5.5. Prediction samples of the 'unify by absolute subtraction' model on the validation set. Rows are illustrating true-positive, false-positive, true-negative, and false-negative prediction instances respectively.

| | Training Both Components | | Traning Second Component Only | |
|---|---|---|---|---|
| | Unify by concatenation | Unify by absolute subtraction | Unify by concatenation | Unify by absolute subtraction |
| Epoch | 6 | 7 | 21 | 112 |
| Training Accuracy | 99.93 % | 99.98 % | 95.58 % | 99.56 % |
| Validation Accuracy | 72.16 % | 70.37 % | 77.67 % | 74.85 % |

Table 5.2. The highest validation accuracy of each model during the training process with corresponding epoch number and training accuracy.

In addition, an experiment is carried out to analyze the performance of the pre-trained ResNet-50 model in extracting the feature-maps from the input image pairs. Since these feature-maps are used to detect any changes in the streets' drivable area, extracting relevant feature-maps from input images has a critical impact on the final performance of the model. The extracted feature-maps are unified to a single feature-map before being fed into the second component. The 'unify by concatenation' model concatenates the feature-maps, and the 'unify by absolute subtraction' model calculates the absolute value of the subtracted feature-maps. The unified feature-map, then, is fed into the second component as an input to be classified as if there is any change in the drivable area of the street or not. As a result, there must be differences between the feature-maps that indicate a change and the feature-maps that indicate no change in the drivable area of the road. As discussed in the methods chapter, the Principal Component Analysis method can be used to reduce the dimensionality of a vector by extracting the principal components while preserving most of the information. Hence, the PCA method is used to extract and plot the principal components of the concatenated feature-maps to illustrate their differences. The PCA plots are illustrated in Figures 5.6 and 5.7.

The PCA plot of figure 5.6 shows the principal components of the feature-maps that are extracted by the ResNet-50 model and unified by concatenation. The feature-maps that show a change in the streets' drivable area tend to appear on the left side of the figure. However, the principal components of the feature-maps that show no change are spread all over the plot. That gives a hint that the pre-trained ResNet-50 model could extract feature-maps from 'change' labeled input images that tend to be on the left side of the PCA plot. However, they are not very different from all feature-maps of the 'no change' labeled input pair images. The PCA plot of figure 5.7 shows the principal components of the feature-maps that are unified by calculating the absolute value of the subtracted feature-maps. As the PCA plot visualizes, the principal components of the

feature-maps are spread over the plot regardless of the fact that the feature-maps show a change in the drivable area of the streets or not. So, it seems that concatenating the extracted feature-maps might be a better approach to unify the feature-maps than calculating the absolute value of the subtracted feature-maps.

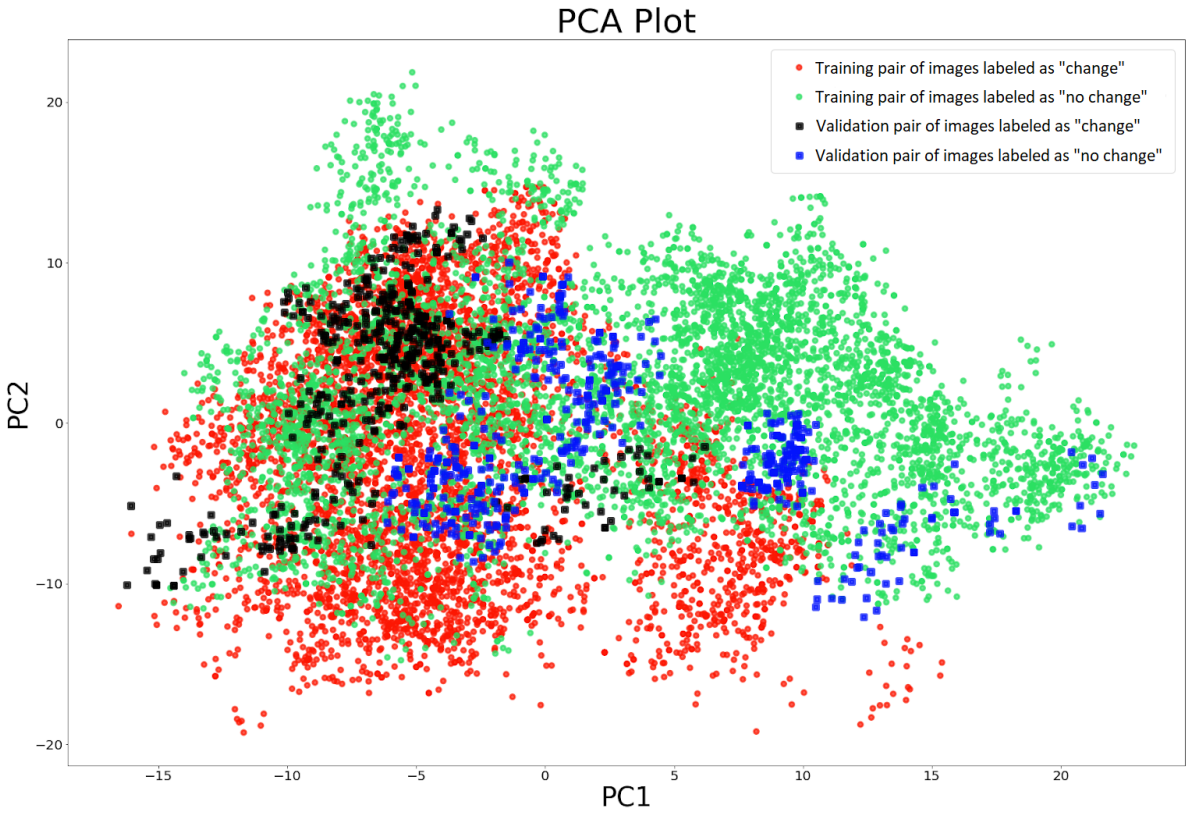



Figure 5.6. PCA plot of the feature-maps that are extracted by pre-trained ResNet-50 model and unified by concatenation. Red and green colored points represent the concatenated feature-maps of the train set image pairs with ‘change’ and ‘no change’ labels, respectively. Black and blue colored points represent the concatenated feature-maps of the validation set image pairs with ‘change’ and ‘no change’ labels, respectively.

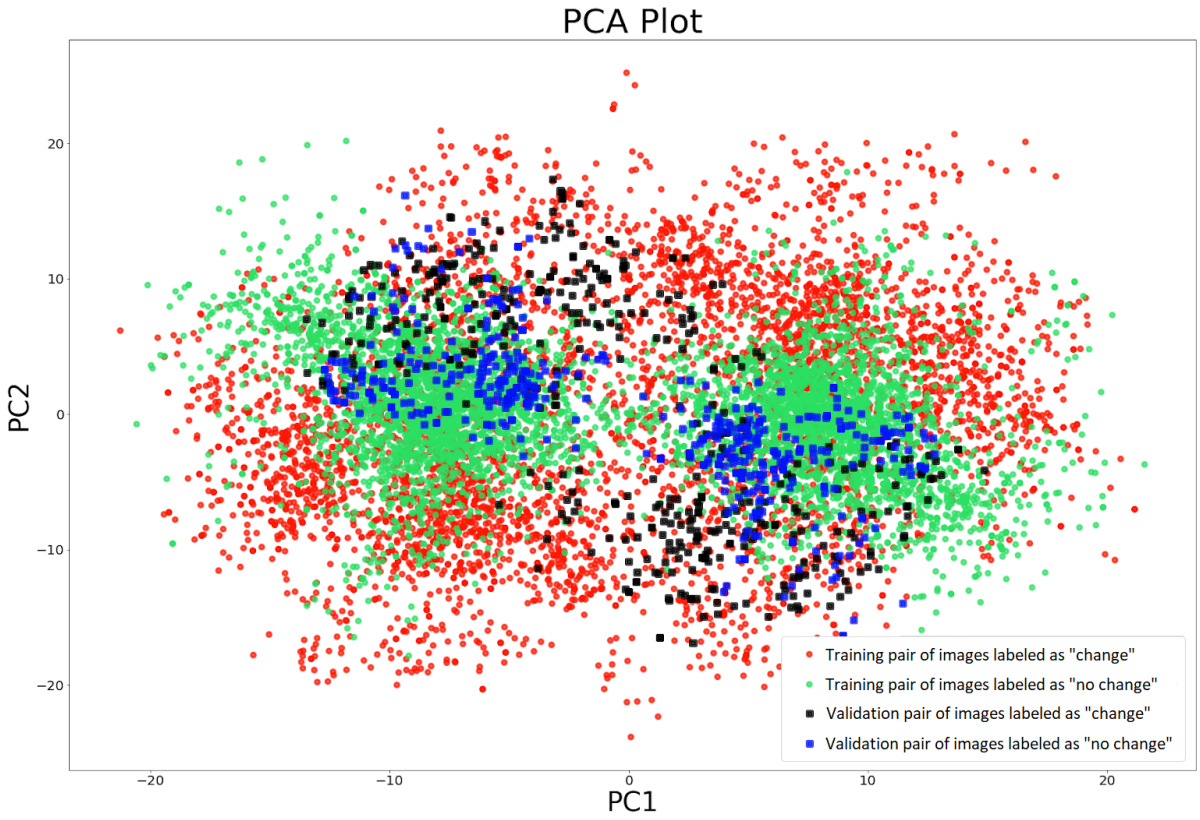



Figure 5.7. PCA plot of the feature-maps extracted by pre-trained ResNet-50 model and unified by calculating the absolute value of the subtracted feature-maps. Red and green colored points represent the unified feature-maps of the train set image pairs with ‘change’ and ‘no change’ labels, respectively. Black and blue colored points represent the unified feature-maps of the validation set image pairs with ‘change’ and ‘no change’ labels, respectively.

# 5.2. Concatenating Image Pairs

Using the pre-trained ResNet-50 model to extract the feature-maps from the input pair images, concatenating them, and training the second component with concatenated feature-maps showed the best performance in experiment 5.1. However, the PCA plot of figure 5.6 shows that extracting feature-maps using a pre-trained ResNet-50 model and concatenating them might not be the best approach for this problem. The PCA plot shows that the feature-maps of the ‘change’ class and the feature-maps of the ‘no change’ class might not differ from each other. If there is no difference between the concatenated feature-maps of two different classes, the second component of the pipeline will not be able to classify the concatenated feature-maps with good performance. Therefore, this experiment concatenates the input pair images before extracting the feature-maps. This way, we can test if the first component can extract feature-maps with more relevant features. More relevant features mean that the extracted feature-maps from image pairs of each class will differ from the extracted feature-maps from image pairs of the other class. To examine this hypothesis, the following model structures are tested.

In the proposed pipeline, the first component takes in two 3-channel images and outputs two feature-maps of size 2048. This experiment aims to eliminate the feature-maps concatenation after extraction and concatenate the input images instead. The first component takes in two images of size 224x224 pixels, which each have three channels for RGB values of image pixels. After that, these two images are concatenated to a single image of size 224x224, but six channels instead. I.e., the channels of the input images are concatenated and resulted in a 6-channel image of size 224x224 pixels. The feature-map of the concatenated image has to be extracted now. The pre-trained ResNet-50 model is used to extract the feature-maps in the proposed pipeline. The ResNet-50 architecture, however, has a fixed input shape of a 3-channel Image of size 224x224 pixels. Two different approaches are used to make the 6-channel image compatible with the ResNet-50 model. One approach is to modify the input layer of the ResNet-50 architecture so that it takes in a 6-channel Image of size 224x224. The output size of the ResNet-50 model, however, remains the same as a vector of size 2048. Therefore, the first component takes in two images and outputs a single feature-map of size 2048. This approach is referred to as ‘modified ResNet’. In this approach, the modified ResNet-50 model cannot use the pre-trained weights of the original ResNet-50 model as the architecture is altered. So, the weights of the modified ResNet-50 model will be initialized and optimized during the training process. The other approach to make the 6-channel image compatible with the input layer of the ResNet-50 model is to use a converter. This converter is a single convolutional layer with an input size of (224,224,6). It has three kernels of size (7,7,6) with a stride value of 1. The

output size of this convolutional layer is (224,224,3), which is compatible with the input layer of the ResNet-50 model. Thus, the first component in this experiment takes in two images of size 224x224x3, concatenates them to an image of size 224x224x6. Then, the trainable single convolutional layer converts the image of size 224x224x6 to 224x224x3. This image is then fed into the ResNet-50 model with pre-trained weights. Accordingly, the output of the first component is a single feature-map of size 2048. This approach is referred to as '6 to 3 channel conversion'. The weights of the ResNet-50 model in this approach are made frozen during the training process.

The extracted feature-map of size 2048 from the concatenated input images goes into the second component as input for detecting whether there is a change in the drivable area of the street or not. The architecture of the second component is the same as in experiment 5.1, but the hyper-parameters are different. The CDIP dataset is used for training the models. The optimizer is the Adam optimizer with a learning rate of 1e-5. The architecture and hyper-parameters of the second component are briefed in table 5.3. And the architecture of the models is provided in figures 5.8 and 5.9.

| Layers | Size | Activation | L2 Regularization | Dropout Rate |
|---|---|---|---|---|
| Input | 2048 | | | |
| Dropout | | | | 0.3 |
| Dense | 100 | ReLU | 0.01 | |
| DropOut | | | | 0.3 |
| Output | 2 | Softmax | | |

Table 5.3. Network architecture and hyperparameters of the second component.

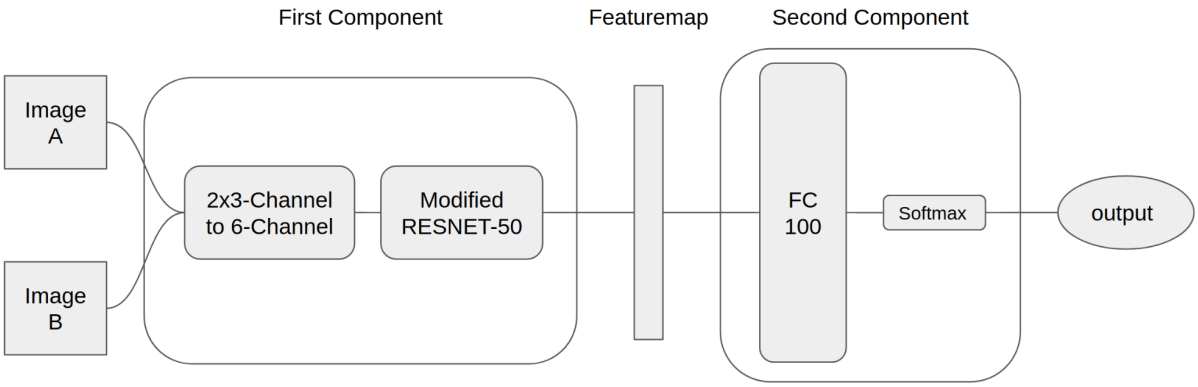



Figure 5.8. The full architecture of the drivable area change detection model. This model uses the 'modified ResNet' approach to make the concatenated 6-channel image compatible with the input layer of the ResNet-50 model. The input images are concatenated and created a single 6-channel image. The 6-channel image is then fed into a modified ResNet-50 model that takes in a 6-channel image as input.

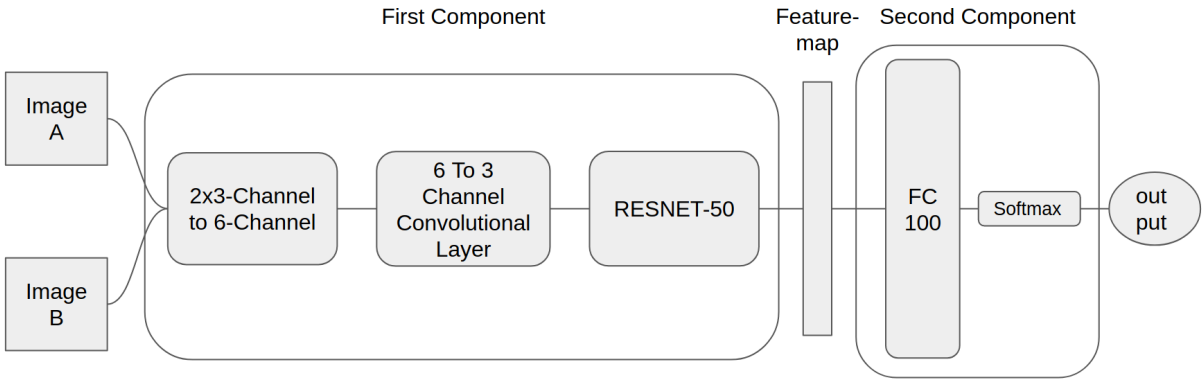



Figure 5.9. The full architecture of the drivable area change detection model. This model uses the '6 to 3 channel conversion' approach to make the concatenated 6-channel image compatible with the input layer of the ResNet-50 model. The input images are concatenated and created a single 6-channel image. The 6-channel image is then fed into a single convolutional layer that converts the 6-channel image to a 3-channel image. Then, the 3-channel image is fed into the pre-trained ResNet-50 model.

Both models are trained for 40 epochs. The training process is stopped after 40 epochs because the validation losses were not decreasing and the validation accuracies were not increasing anymore. The training history plots are illustrated in figure 5.10. This plot shows that concatenating the input images instead of concatenating the extracted feature-maps decreased the performance of the model. The best validation accuracy in experiment 5.1 is 77.66%. The drivable area change detection model in this experiment could achieve the validation accuracy of 68.91%. The best performance of each model is presented in table 5.4.

| | Modified ResNet Approach | 6 to 3 Channel Conversion Approach |
|---|---|---|
| Epoch | 36 | 4 |
| Training Accuracy | 99.74 % | 86.26 % |
| Validation Accuracy | 53.31 % | 68.91 % |

Table 5.4. The highest validation accuracy of each model during the training process with corresponding epoch number and training accuracy.

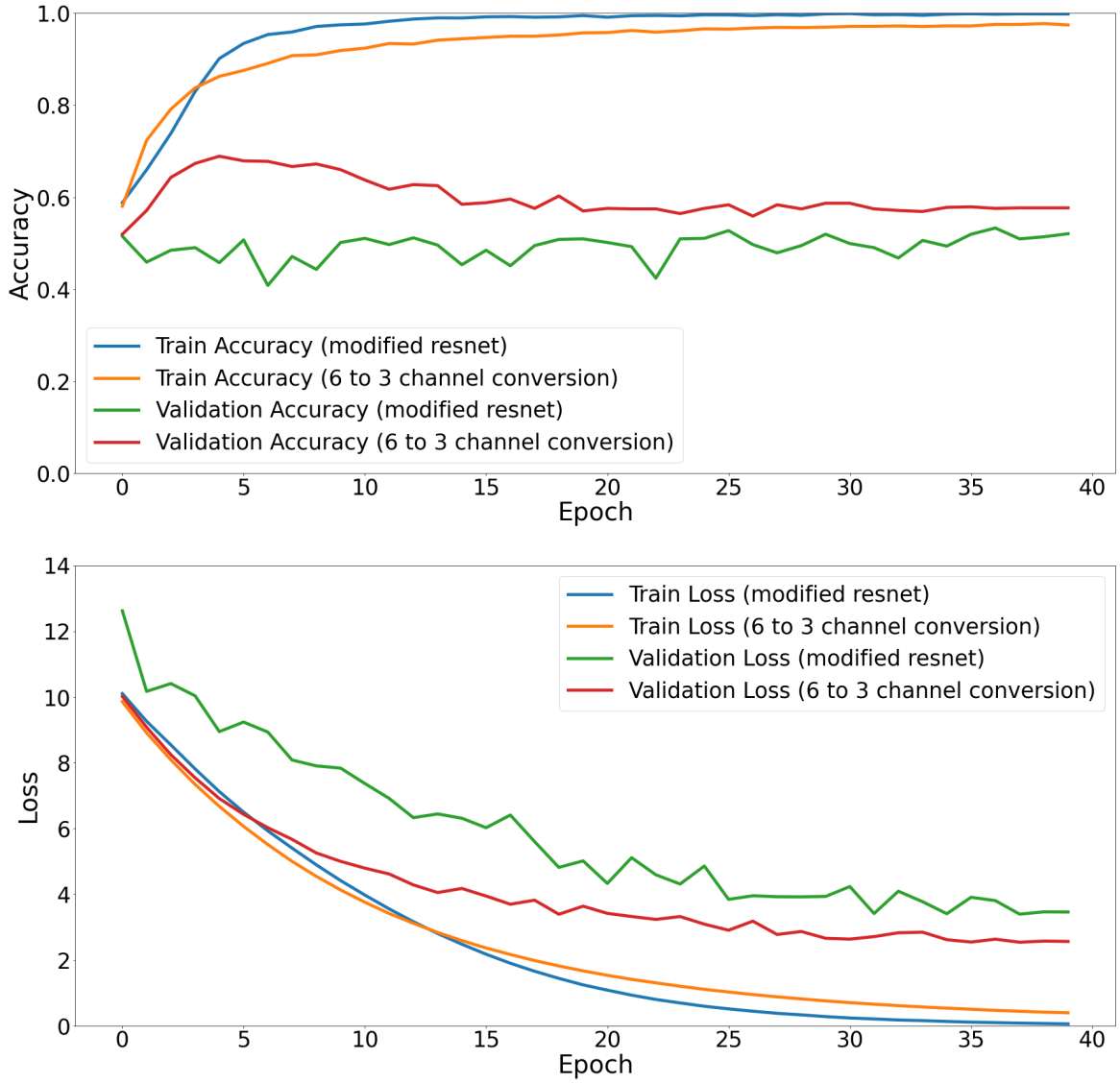



Figure 5.10. Training history of the two models. One of the models is using the 'modified ResNet' approach to make the concatenated 6-channel image compatible with the ResNet-50 model. The other model uses the '6 to 3 channel conversion' approach. As the plots show, both models suffer extreme overfitting on the train set.

# 5.3. Improving Second Component

Considering the previous experiments, it seems that the best architecture so far for the first component is to use the pre-trained ResNet-50 model to extract the feature-maps from input pair images. Based on experiment 5.1, the best approach to unify the extracted feature-maps in this work is to concatenate them. The second component of the proposed pipeline is a feed-forward neural network. The input of this component is the unified feature-maps extracted from input pair images by the first component. The second component is responsible for classifying the unified feature-maps as if there is any change in the street's drivable area of the input pair images. This subsection discusses the experiment to enhance the performance of the second component in classifying the unified feature-maps.

Here, the first component is the same as in experiment 5.1. The pre-trained ResNet-50 model is used to extract the feature-maps from input pair images as the first component. The feature-maps are extracted and stored in advance. They are loaded, concatenated, and fed into the second component as an input during the training process of the second component. The second component in this experiment is composed of an input layer of size 4096, two dense layers of sizes 50 and 20 with ReLU activation function and L2 regularization, and an output layer. The output layer is a layer of size 2 with the softmax activation function. There are three dropout layers in the architecture of the second component. The dropout layers are listed in table 5.5. The artificial neural network model weights are optimized by the Adam optimizer with a learning rate of 1e-06. The model is trained by the LCDIP dataset in this experiment. The architecture of the full model in this experiment is provided in figure 5.11.

| Layers | Size | Activation | L2 Regularization | Dropout Rate |
|---|---|---|---|---|
| Input | 4096 | | | |
| Dropout | | | | 0.3 |
| Dense | 50 | ReLU | 0.05 | |
| DropOut | | | | 0.3 |
| Dense | 20 | ReLU | 0.05 | |
| DropOut | | | | 0.3 |
| Output | 2 | Softmax | | |

Table 5.5. Network architecture and hyperparameters of the second component.

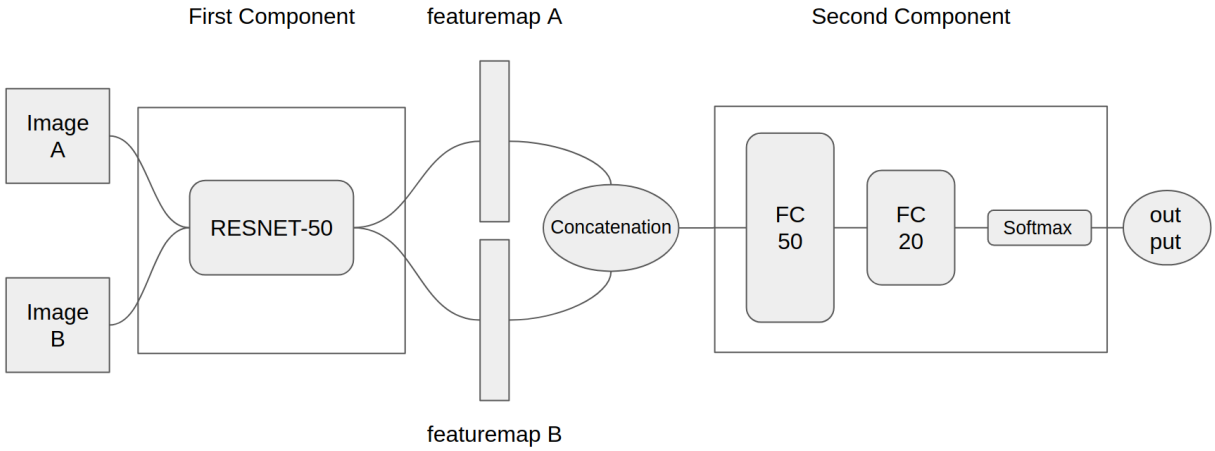



Figure 5.11. Full Model Architecture of the Experiment

The model is trained for 200 epochs. The training process was stopped after 200 epochs because the validation loss stopped decreasing and the validation AUC stopped increasing. The training history plot is illustrated in figure 5.12. As the plot shows, there is overfitting on the train set from early epochs. The overfitting rate is increasing as the number of training epochs increases. However, the performance of the drivable area change detection model is increased slightly compared to experiment 5.1. The best validation accuracy in experiment 5.1 is 77.67%. The validation accuracy in this experiment is 79.52%. This means altering the second component's architecture increased the performance of the model by 2.38%. The model accuracies are provided in table 5.6. And figure 5.13.

| | Validation Accuracy | Validation AUC |
|---|---|---|
| Best Performance in Experiment 5.1 | 77.67 % | - |
| Current Experiment Performance | 79.52 % | 88.91 % |

Table 5.6. The highest validation accuracy achieved by the drivable area change detection model in experiment 5.1 is 77.67%. The drivable area change detection model in this experiment could achieve a validation accuracy of 79.52%. This shows a 2.38% improvement in the model's performance.

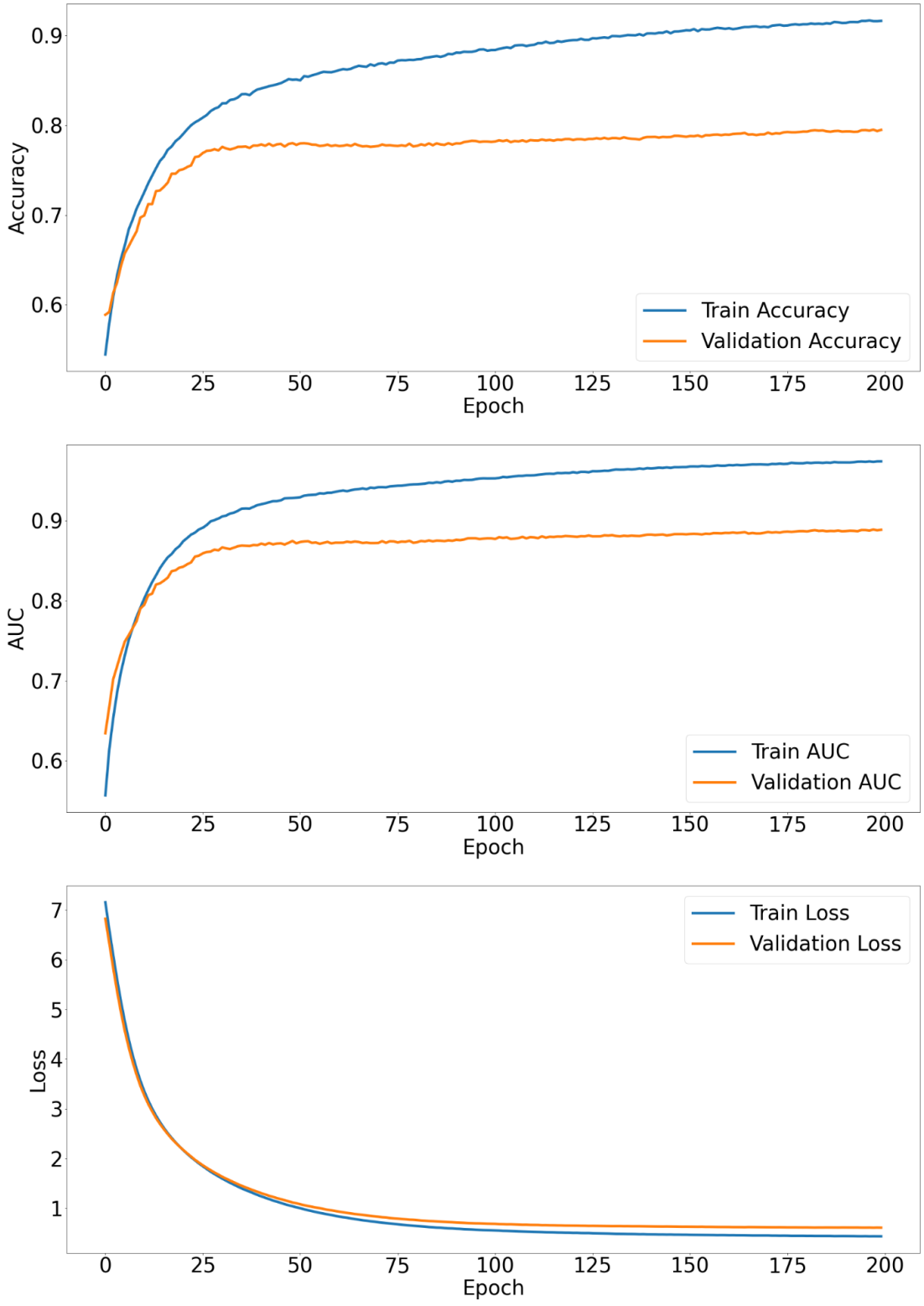



Figure 5.12. The training history plot of the drivable area change detection model.

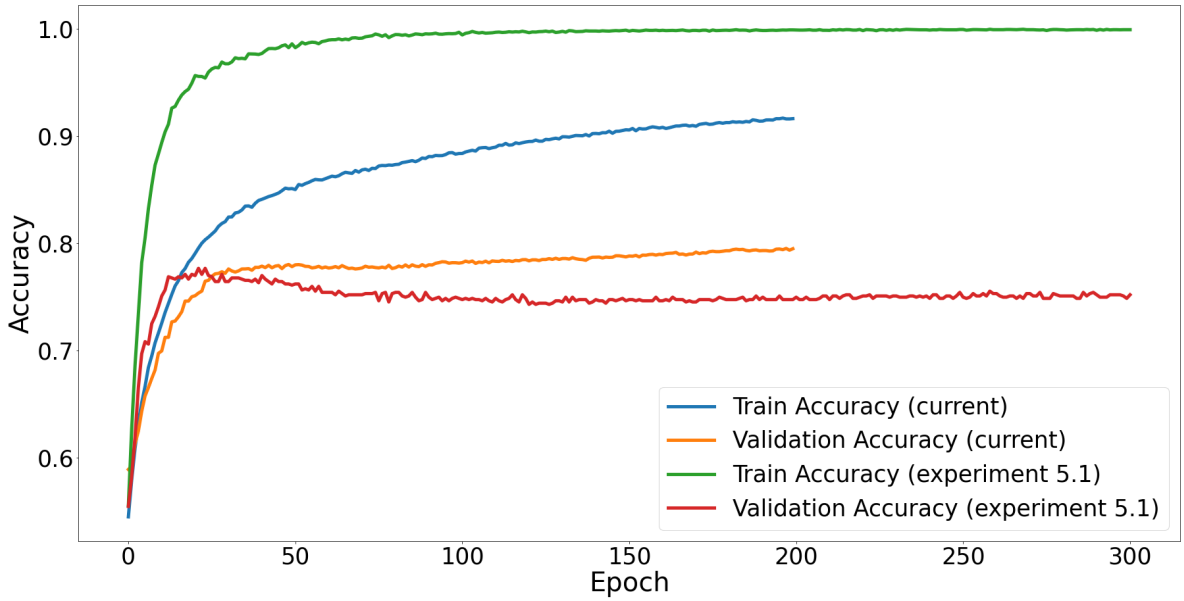
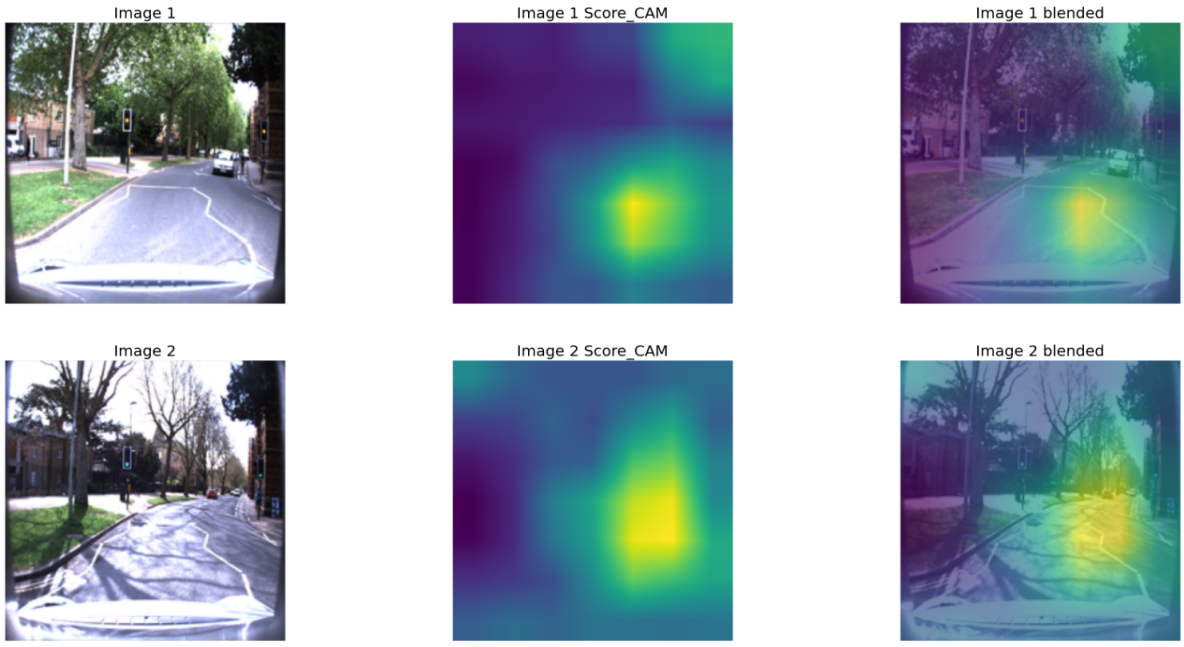


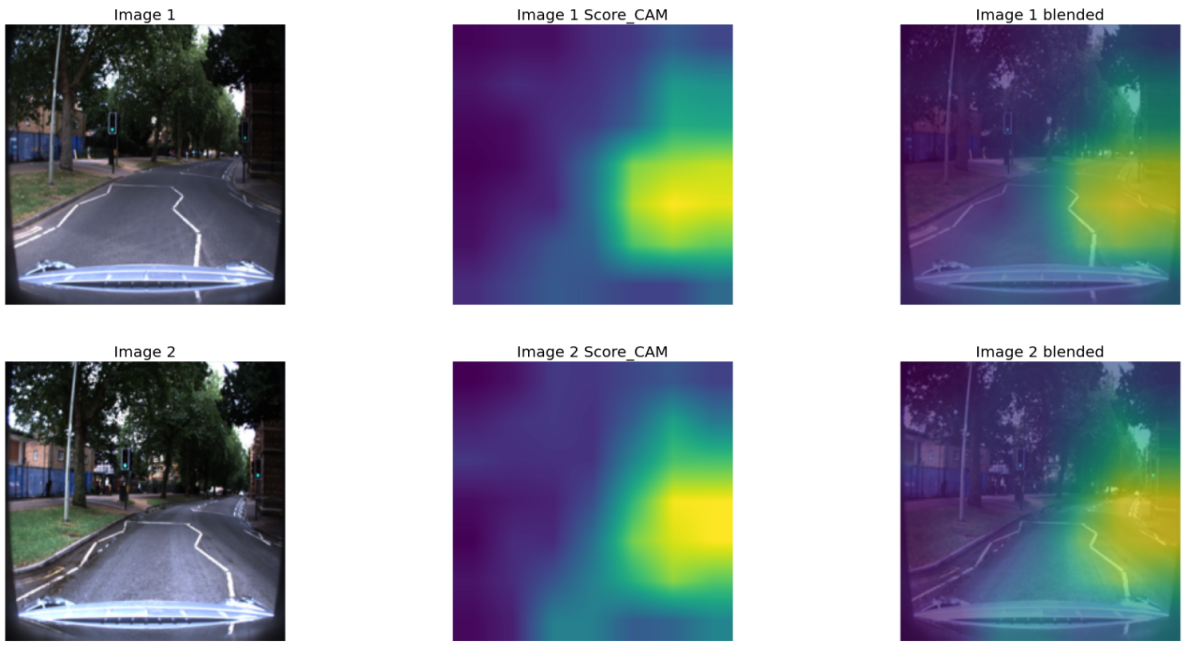

Figure 5.13. Comparing the training history of the model that has the highest validation accuracy in experiment 5.1 with the model's training history of the current experiment.

The performance of the drivable area change detection model is improved by 2.38% in this experiment. We decided to evaluate the model's performance by generating and analyzing the Score-CAM images. Score-CAM images are generated to visualize the regions of the input images that the model focuses on the most. Few of these Score-CAM images are presented in the following figures. As it is illustrated in figures, in a few cases, the neural network focuses on the correct region of the image and outputs the correct class. But, in most cases, the area that the network focuses on does not seem to be proper for classifying the change in the drivable area of the streets. For example, sometimes, the network focuses on the sky or the buildings to detect the change in the road's drivable area, which is not the right region of the image to focus on.

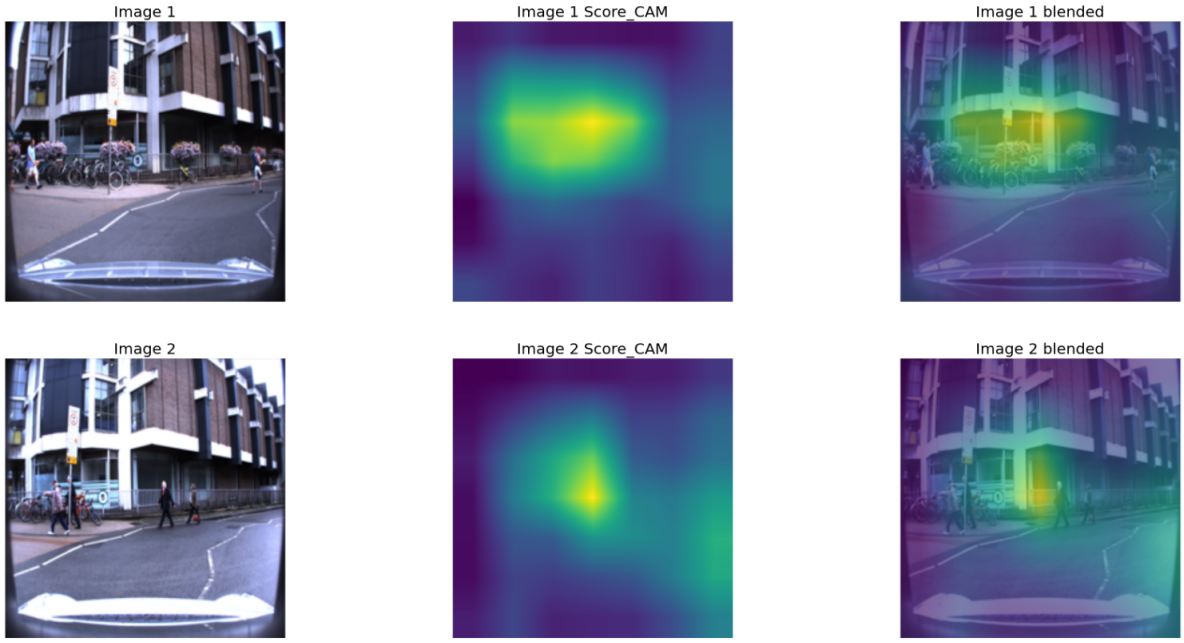
y_true:'no change'  y_pred:'no change'  P('no change'):0.91  P('change'):0.09



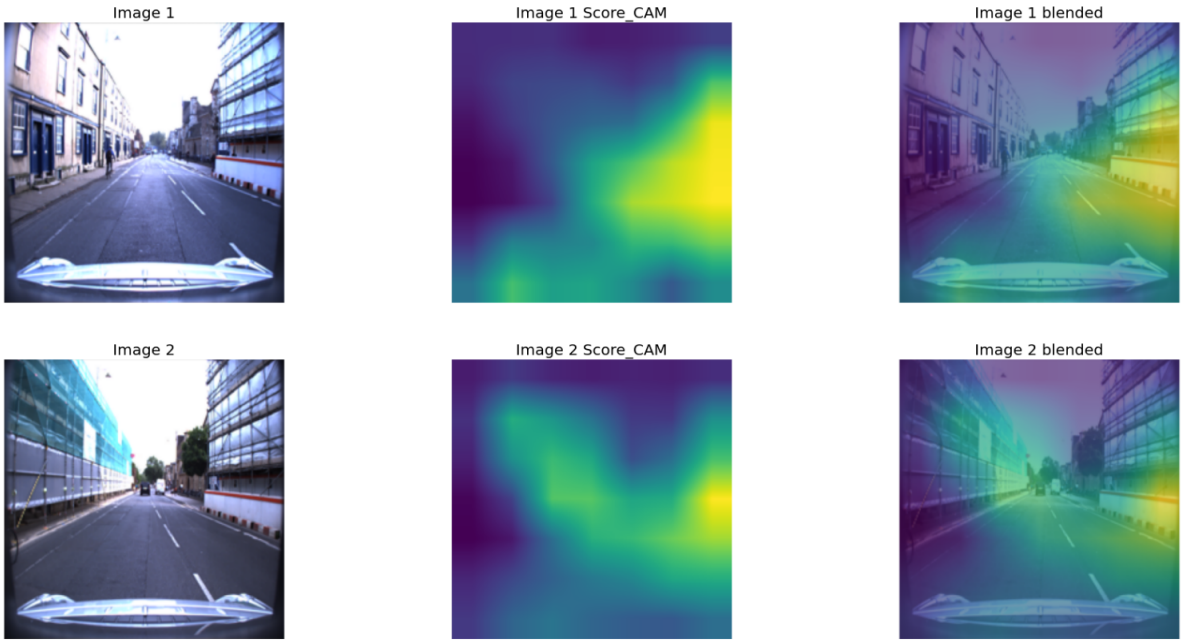

y_true:'no change'  y_pred:'no change'  P('no change'):0.92  P('change'):0.08



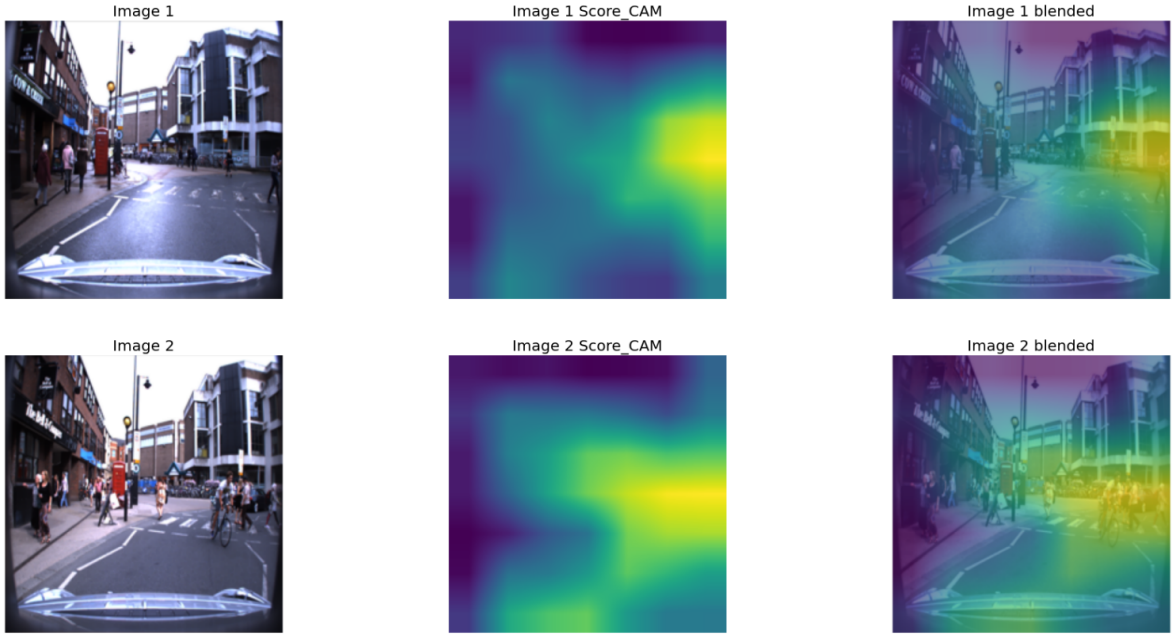
y_true:'no change'  y_pred:'change'  P('no change'):0.27  P('change'):0.73



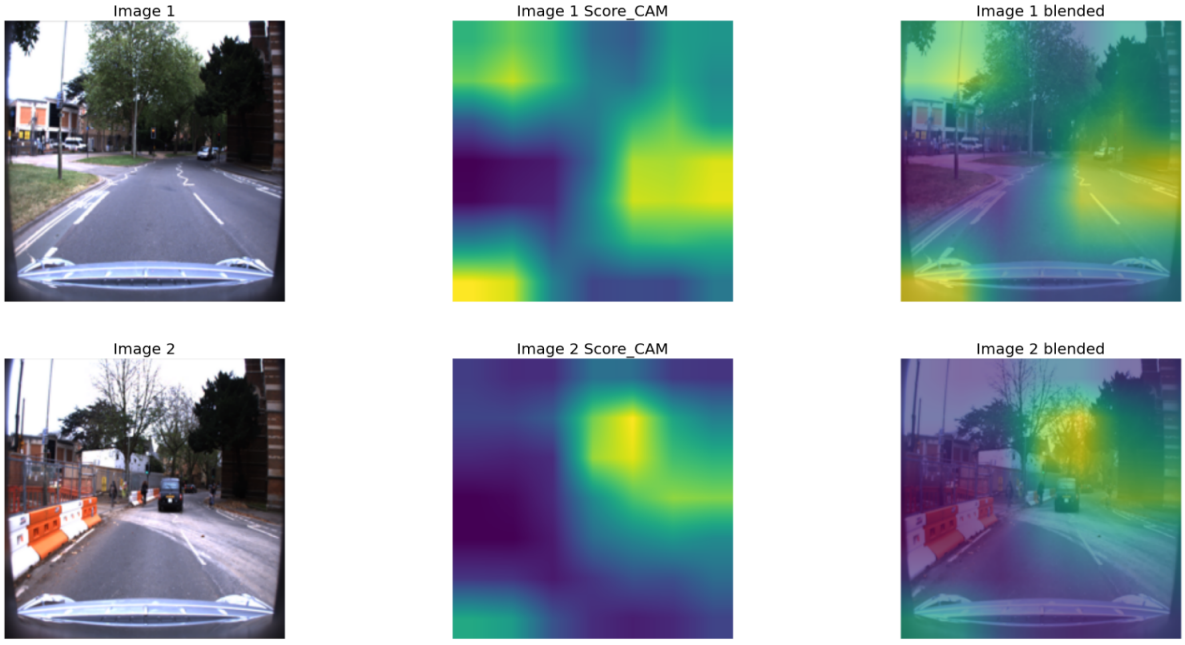

y_true:'no change'  y_pred:'change'  P('no change'):0.47  P('change'):0.53



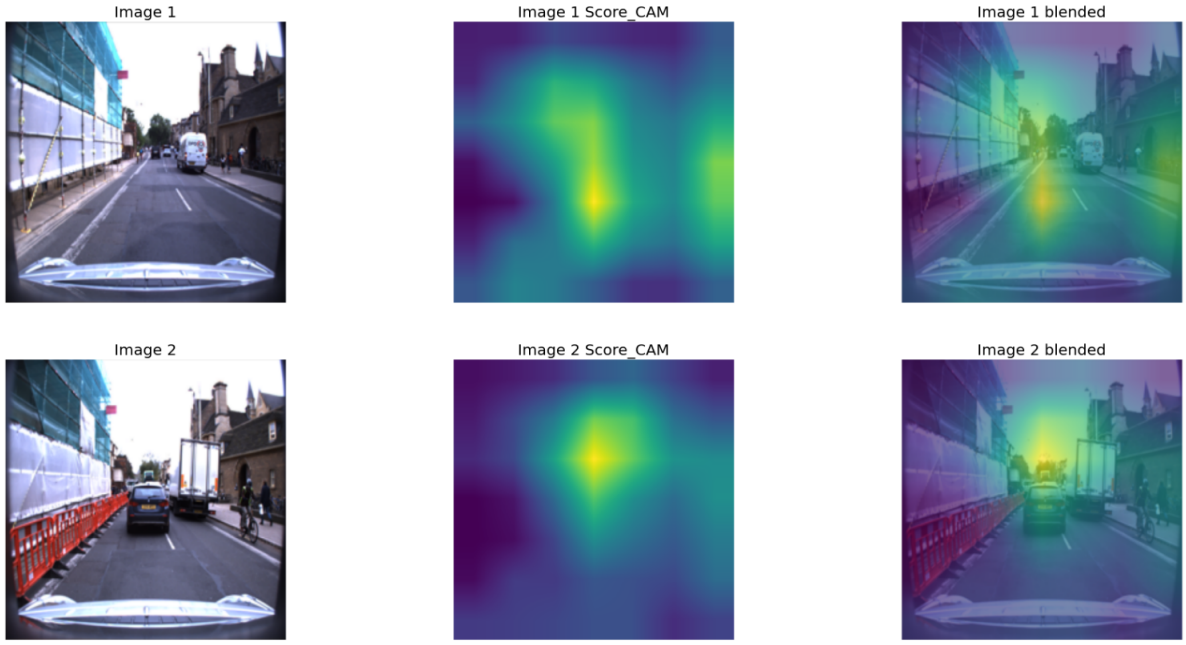
y_true:'no change'  y_pred:'change'  P('no change'):0.28  P('change'):0.72



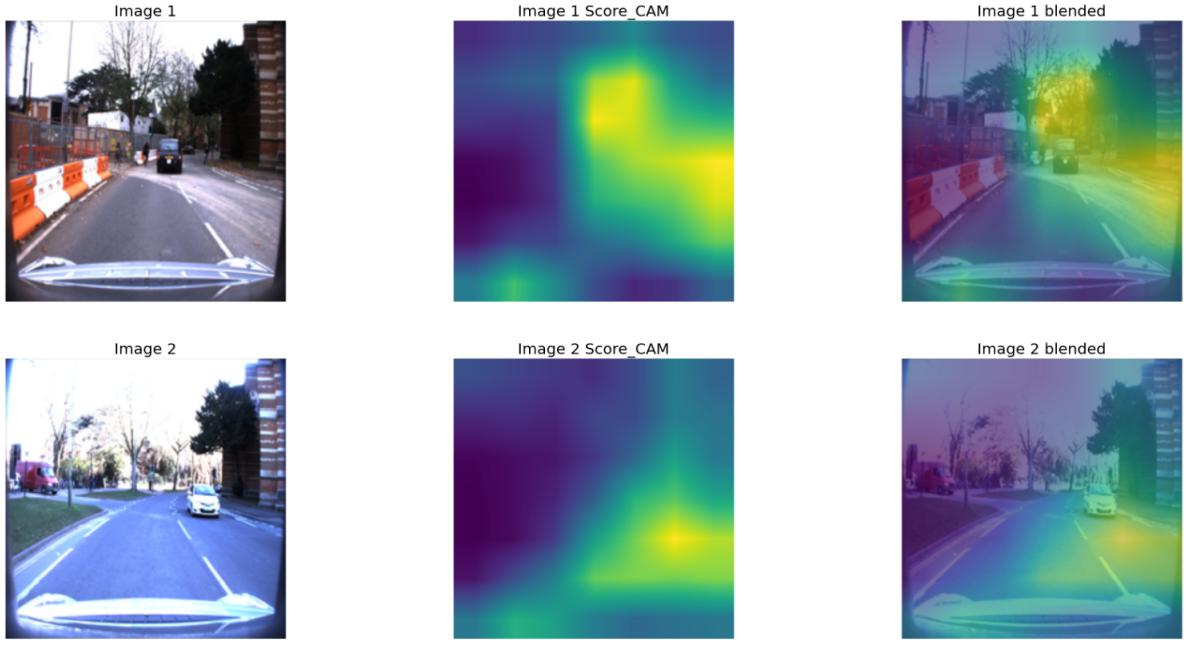

y_true:'change'  y_pred:'no change'  P('no change'):0.54  P('change'):0.46



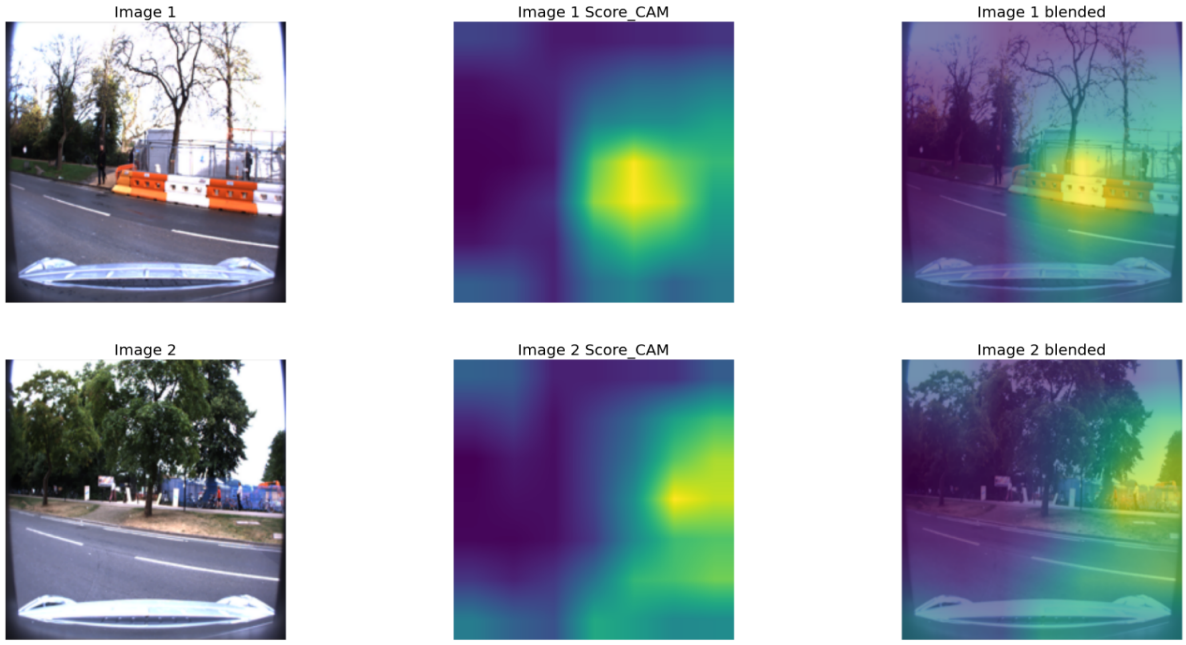
y_true:'change'  y_pred:'no change'  P('no change'):0.51  P('change'):0.49



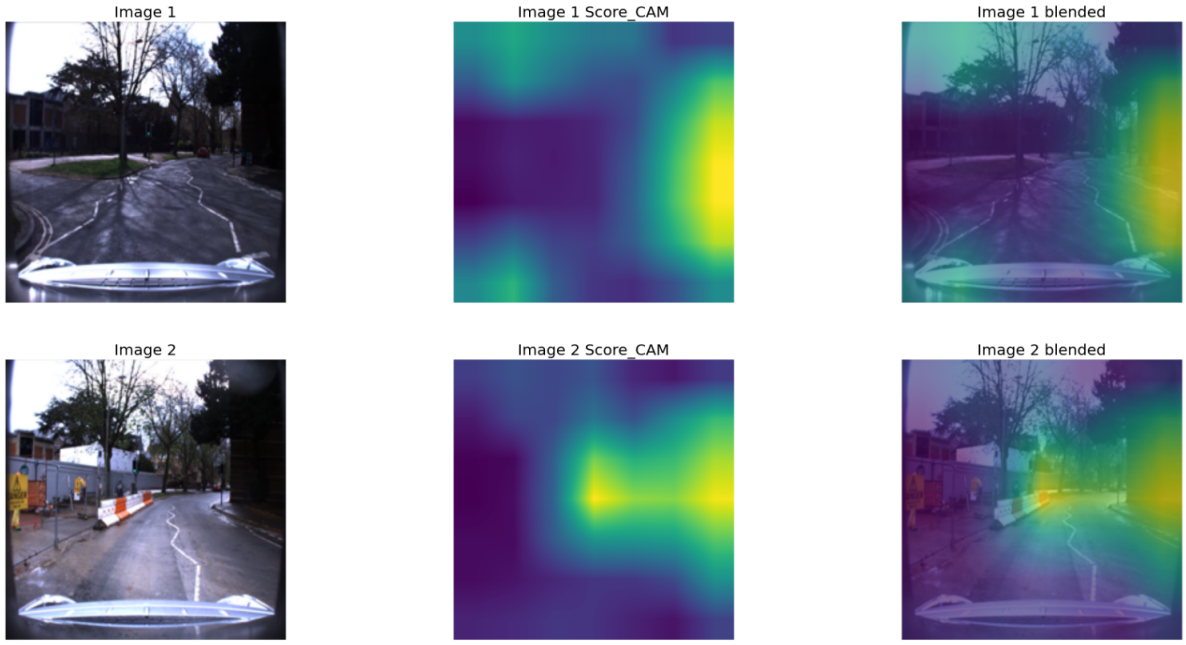

y_true:'change'  y_pred:'no change'  P('no change'):0.62  P('change'):0.38

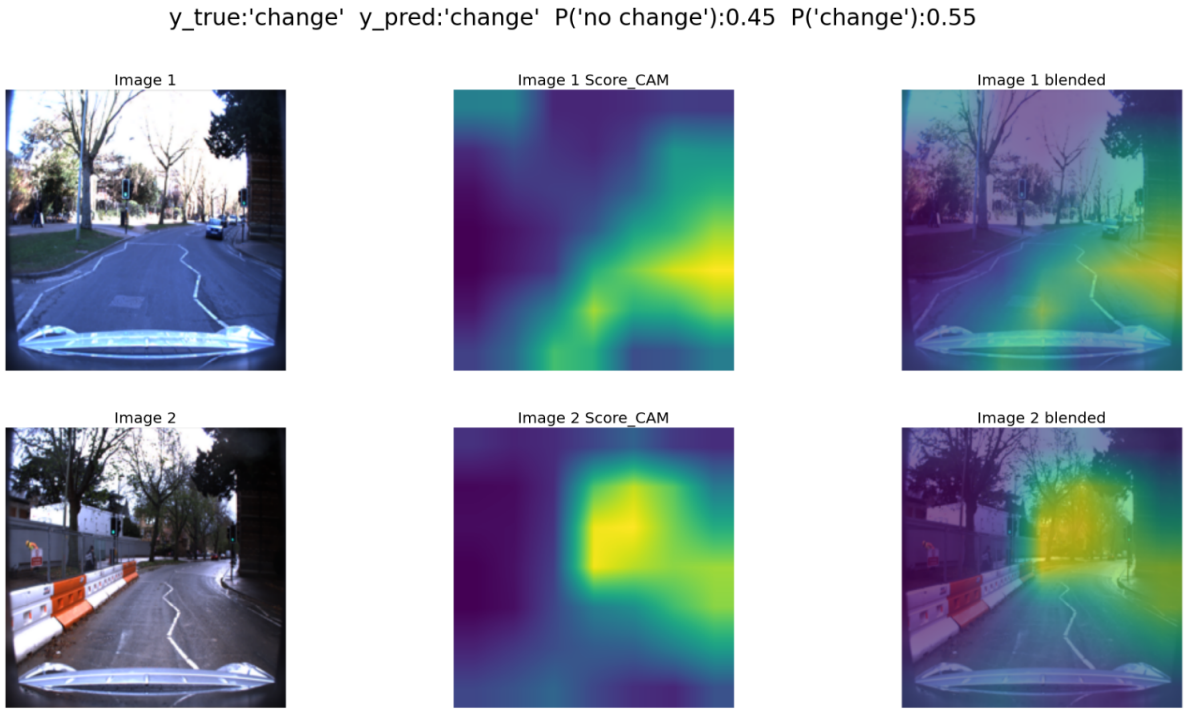


y_true:'change'  y_pred:'change'  P('no change'):0.37  P('change'):0.63




y_true:'change'  y_pred:'change'  P('no change'):0.38  P('change'):0.62






In addition, a cross-validation experiment is carried out to evaluate the generalization capacity of the model further. This experiment analyzes the generalization capacity of the model by training and evaluating the model with different train/validation splits. Three hundred thirty-five validation folds are generated for this experiment using the approach discussed in the dataset chapter. Adam optimizer and SGD optimizer are used in the cross-validation experiment as the artificial neural network's optimizer. So, each validation fold is used twice. Once for training using the Adam optimizer, and once, using the SGD optimizer. The distribution of the validation AUC scores of the folds is visualized in figure 5.14. The box plot of the model's validation AUC is illustrated in figure 5.15. As figure 5.14 shows, the neural network's AUC values of the validation folds mainly vary from 0.68 to 0.92. According to the plot, the 0.8 AUC value has the most distribution. According to figure 5.15, the mean AUC value of the neural network in the cross-validation experiment is around 0.8. And the network has the AUC measure between 0.76 and 0.83 on half of the validation folds. We can say that the average AUC measure of this experiment's network on the LCDIP dataset is 0.8. In addition, the AUC score distributions in figure 5.14 and the box plot of figure 5.15 show that the final performance of the model with the Adam optimizer is almost the same as the final performance of the model with the SGD optimizer. The AUC scores have almost identical distributions, mean, and median values.

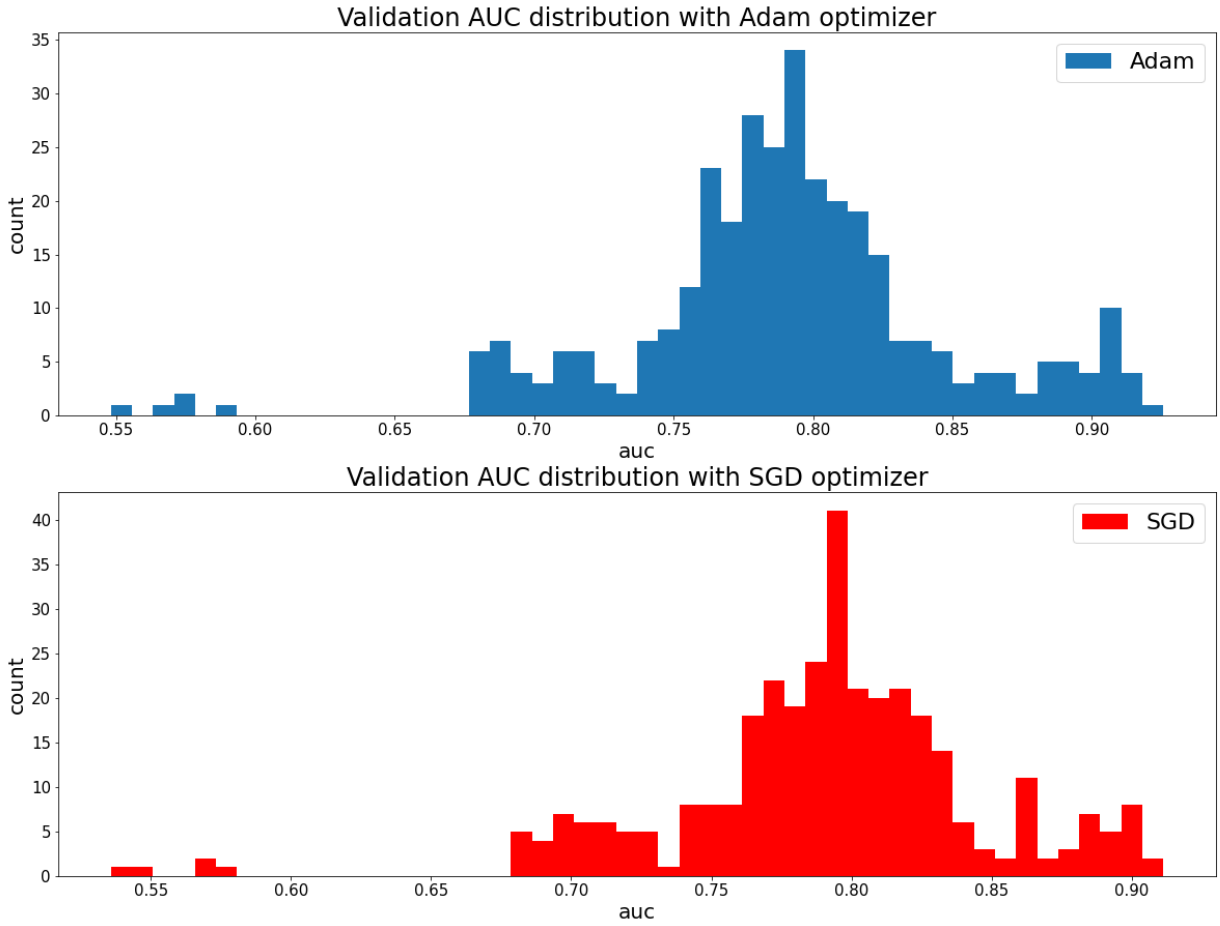



Figure 5.14 Validation AUC score distribution of the drivable area change detection model in the cross-validation experiment with Adam and SGD optimizers. As the distribution plots show, using the SGD optimizer or Adam optimizer does not change the AUC score distribution noticeably.

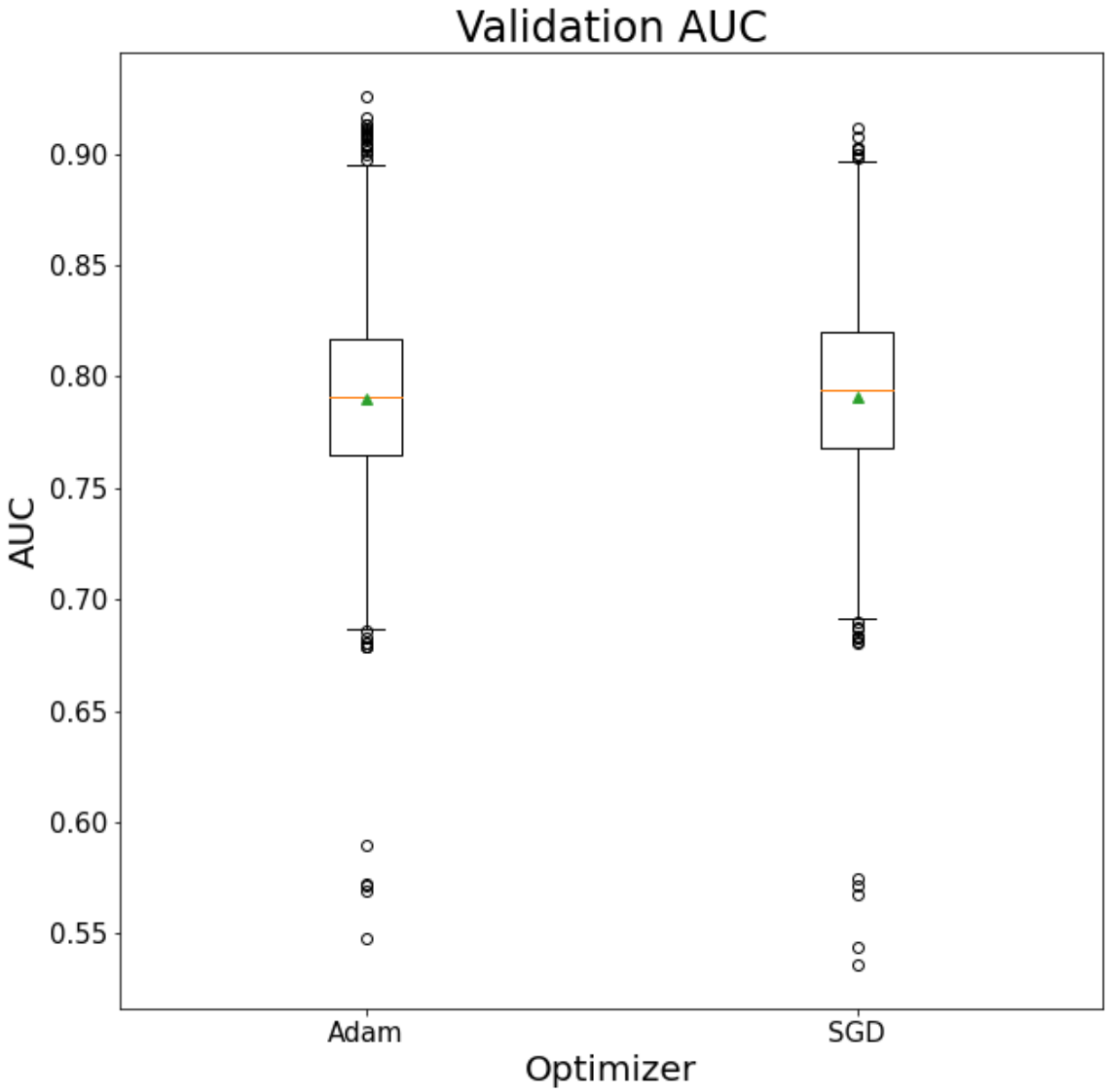



Figure 5.15. Validation AUC score boxplot of the cross-validation experiment with the Adam optimizer and the SGD optimizer.

# 5.4. Fine-Tuning ResNet

Previous subsections discussed some experiments to find an optimal neural network architecture and hyperparameters for the drivable area change detection problem. The neural network pipeline consists of two components. Using the pre-trained ResNet-50 model to extract the feature-maps from input pair images showed the best performance so far. The feature-maps are extracted from input images separately. They are loaded, concatenated, and fed into the second component as input during training of the second component. The best architecture for the second component is a feed-forward multi-layer perceptron network with two hidden layers. As discussed in experiment 5.1, including the weights of the ResNet-50 model (first component) into the training process did not increase the final performance of the drivable area change detection model. Moreover, experiment 5.2 showed that concatenating the input pair images to a single image with 6-channels (RGBRGB) and feeding it into a trainable modified ResNet-50 model did not work well. It hindered the second component from classifying the feature-maps as if there is a change in the drivable area of streets or not. Therefore, this experiment tries to fine-tune the pre-trained weights of the ResNet-50 model separately. And use the fine-tuned ResNet-50 model as the first component of the proposed pipeline. Fine-tuning the pre-trained ResNet-50 model is planned by hoping that the fine-tuned ResNet-50 model extracts more relevant feature-maps from the input images.

This fine-tuning is performed by using the Triplet Loss. To fine-tune a feature-map extractor convolutional neural network with the Triplet Loss, the network needs to have three input images. The first one is used as the reference image and is called the ‘anchor’. In this problem, an image of a street that does not show any changed drivable area of the street must be used as ‘anchor’ image input. The second input is referred to as the ‘same’. In other words, the image of the ‘same’ input is a picture of the same street and the same spot that the drivable area of the road is identical to as in the ‘anchor’. The third input image is referred to as the ‘different’. This input is an image of the same street but with an altered drivable area. As a result, the inputs are the ‘anchor’, the ‘same’, and the ‘different’. The LCDIP dataset is used to prepare a set of these triplet images. The LCDIP dataset contains image pair instances of a street with both ‘change’ (positive) and ‘no change’ (negative) labels. To prepare the triplet image set, all instances of the LCDIP dataset that have positive labels are selected. For each selected positive labeled instance, at least one negative labeled instance shows no change in the drivable area of the same street. Hence, for each positive instance of the LCDIP dataset, a negative instance is selected randomly from the corresponding negative instance/s. As a result, a triplet image set is extracted from each of the selected positive and negative instances of the LCDIP dataset. Such that the anchor is

the image that exists in both positive and negative instances, the ‘same’ image is the other image of the negative instance, and the ‘different’ is the other image of the positive instance. These triplet images are then fed into the ResNet-50 model, the feature-map extractor convolutional neural network that we want to fine-tune. The ResNet-50 model will extract the feature-maps from all input images and output three separate feature-maps of size 2048 corresponding to each input image. These three feature-maps then go to the Triplet Loss for calculating the loss value. The Triplet Loss is calculated as follows.

$$Triplet\ Loss\ = max\left(\left(Dist(anchor, same)\ -\ Dist(anchor, different)\ +\ \alpha\right),\ 0\right)$$

The Triplet Loss function tries to minimize the Euclidean distance between feature-maps of the ‘anchor’ and the ‘same’ input images and maximize the Euclidean distance between feature-maps of the ‘anchor’ and the ‘different’ input images. Here $\alpha$ is the optimal difference between the two distance values. In other words, if the difference of the Euclidean distance of the ‘anchor’ and the ‘same’ and the Euclidean distance of the ‘anchor’ and ‘different’ is more than $\alpha$, the loss would be zero. In this experiment, $\alpha$ is 0.2. Therefore, by fine-tuning the ResNet-50 model using this loss function, the optimizer will try to minimize the loss. This means the optimizer will try to modify the weights of the ResNet-50 model such that the euclidean distance between extracted feature-maps from a street’s images that have the same drivable area decreases. While the euclidean distance between extracted feature-maps from a street’s images that have altered drivable area increases. This fine-tuning may eventually help the second component classify the extracted feature-maps from the input images more precisely. The Adam optimizer with a learning rate of 5e-7 is used for fine-tuning the ResNet-50 model with the Triplet Loss. The diagram of fine-tuning of the ResNet-50 model with the Triplet Loss is provided in figure 5.16. The pre-trained ResNet-50 model is fine-tuned for two epochs using the Triplet Loss.

The fine-tuned ResNet-50 model, then, is used as the first component of the proposed pipeline to extract the feature-maps from the input images. The extracted feature-maps are fed into the second component as input for the classification task. The structure of the second component in this experiment is the same as in experiment 5.3. The Adam optimizer with the learning rate of 1e-6 is used to optimize the network’s weights during the training phase. The LCDIP dataset is used to train the second component. The first component (fine-tuned ResNet-50 model by Triplet Loss) is not included in the training process of the second component. Instead, the feature-maps of the input image pairs are extracted and stored by the first component in advance. They are loaded, concatenated, and fed into the second component as input during the training process.

The architecture of the second component is briefed in table 5.7. And the architecture of the pipeline of this experiment is displayed in figure 5.17.

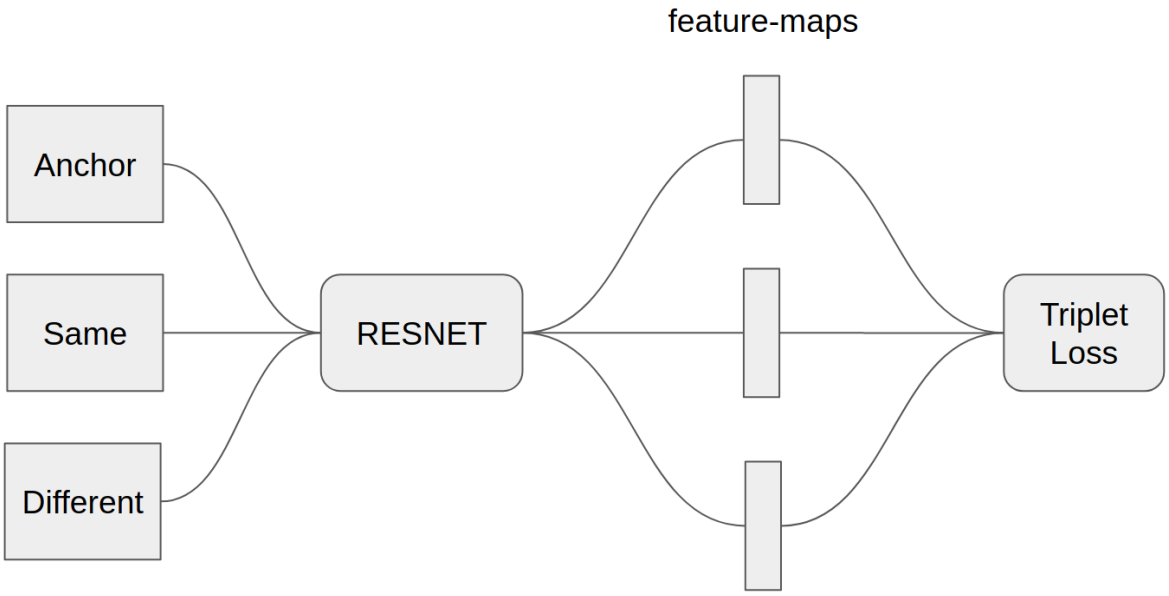



Figure 5.16. The architecture of fine-tuning the pre-trained ResNet-50 model with Triplet Loss.

| Layers | Size | Activation | L2 Regularization | Dropout Rate |
|---|---|---|---|---|
| Input | 4096 | | | |
| Dropout | | | | 0.3 |
| Dense | 50 | ReLU | 0.05 | |
| DropOut | | | | 0.3 |
| Dense | 20 | ReLU | 0.05 | |
| DropOut | | | | 0.3 |
| Output | 2 | Softmax | | |

Table 5.7. Network architecture and hyperparameters of the second component.

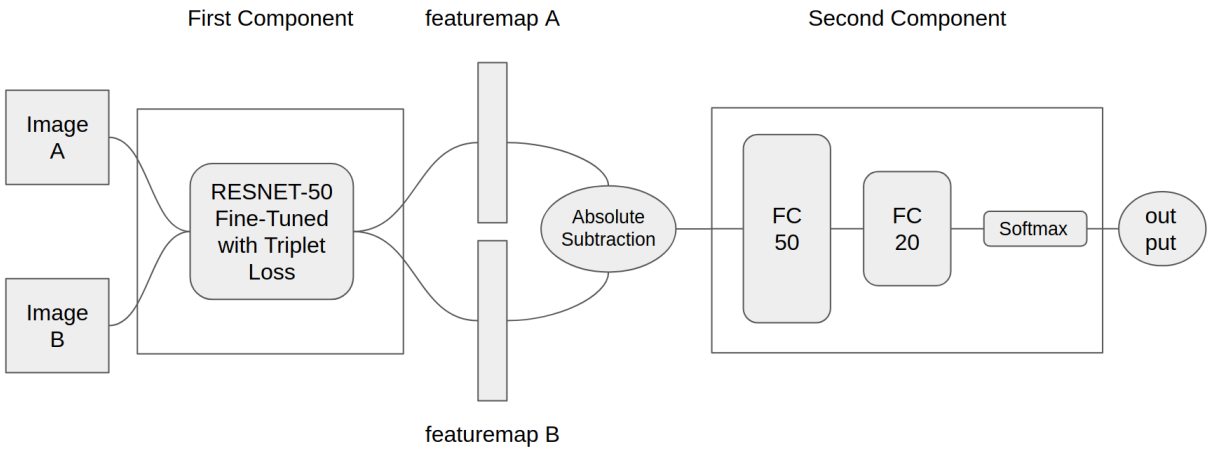



Figure 5.17. The drivable area change detection pipeline.

The second component is trained for 150 epochs. The training process was stopped after 150 epochs because the validation loss did not decrease and the validation AUC did not increase anymore. The training history plot is illustrated in figure 5.18. The training history plot shows that unlike other experiments, the drivable area change detection model did not start to overfit on the training set from the early epochs. The overfitting started around the fiftieth epoch. The validation accuracy and AUC of the model are considerably higher with the fine-tuned ResNet-50 model as the first component. The best validation accuracy in experiment 5.3 is 79.52%. Whereas the validation accuracy in this experiment is 88.26%. Therefore, fine-tuning the pre-trained ResNet-50 model with Triplet Loss increased the performance of the model by 11%. The model accuracies are provided in table 5.8. And figure 5.18.

| | Validation Accuracy | Validation AUC |
|---|---|---|
| Best Performance in Experiment 5.1 | 77.67 % | - |
| Best Performance in Experiment 5.3 | 79.52 % | 88.91 % |
| Current Experiment Performance | 88.26 % | 93.45 % |

Table 5.8. The highest validation accuracy achieved by the drivable area change detection model in experiment 5.3 is 79.52%. The drivable area change detection model in this experiment could achieve a validation accuracy of 88.26%. This shows 11% improvement in the model's performance.

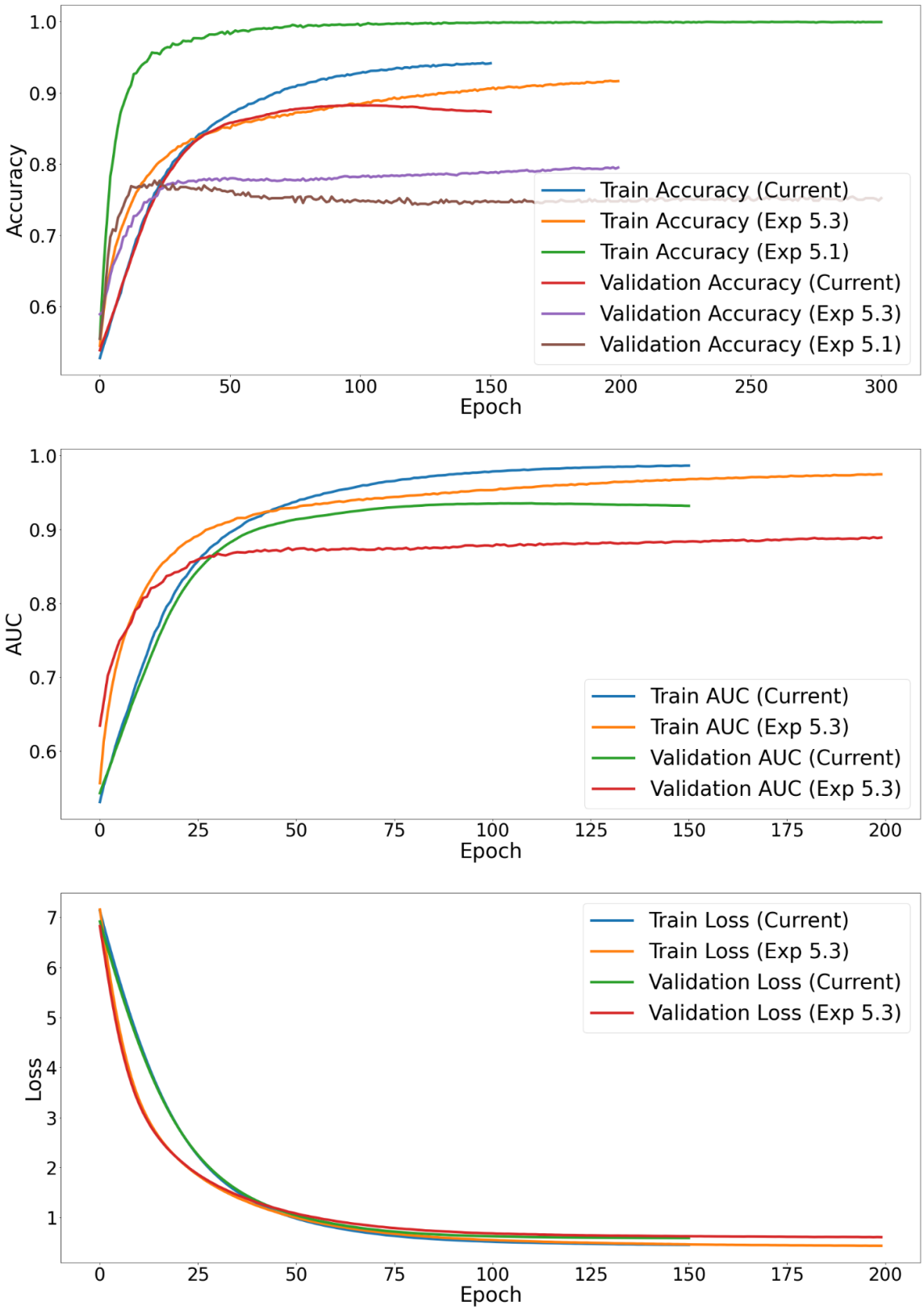
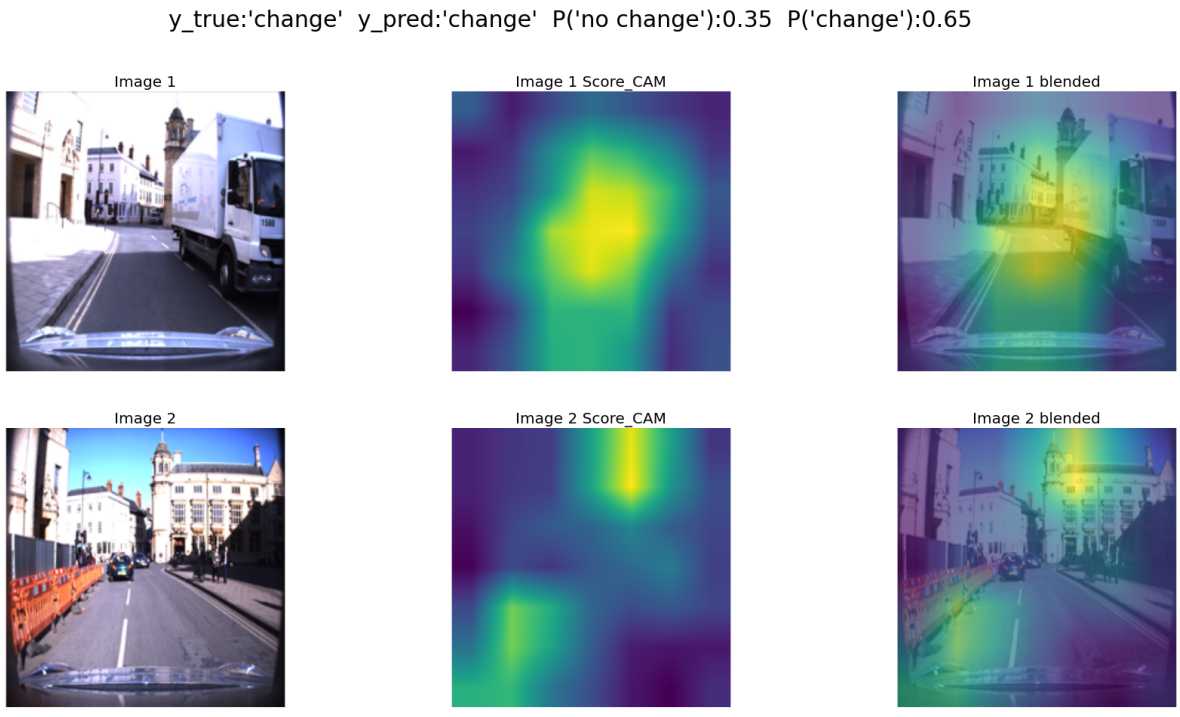
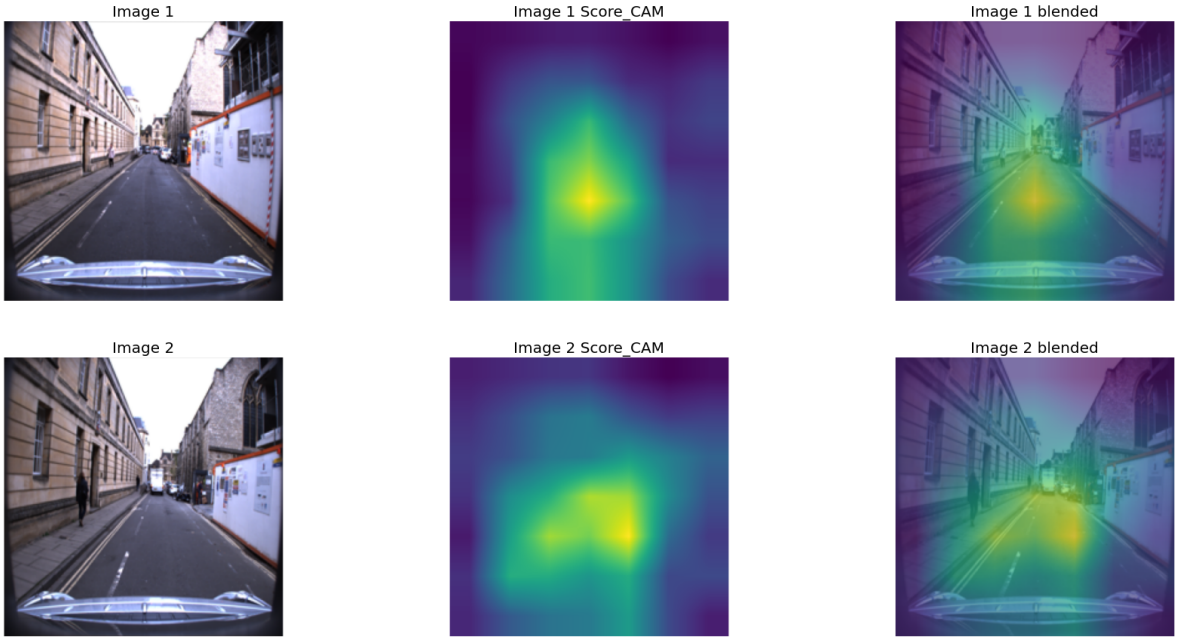



Figure 5.18. The training history plot of the drivable area change detection model.

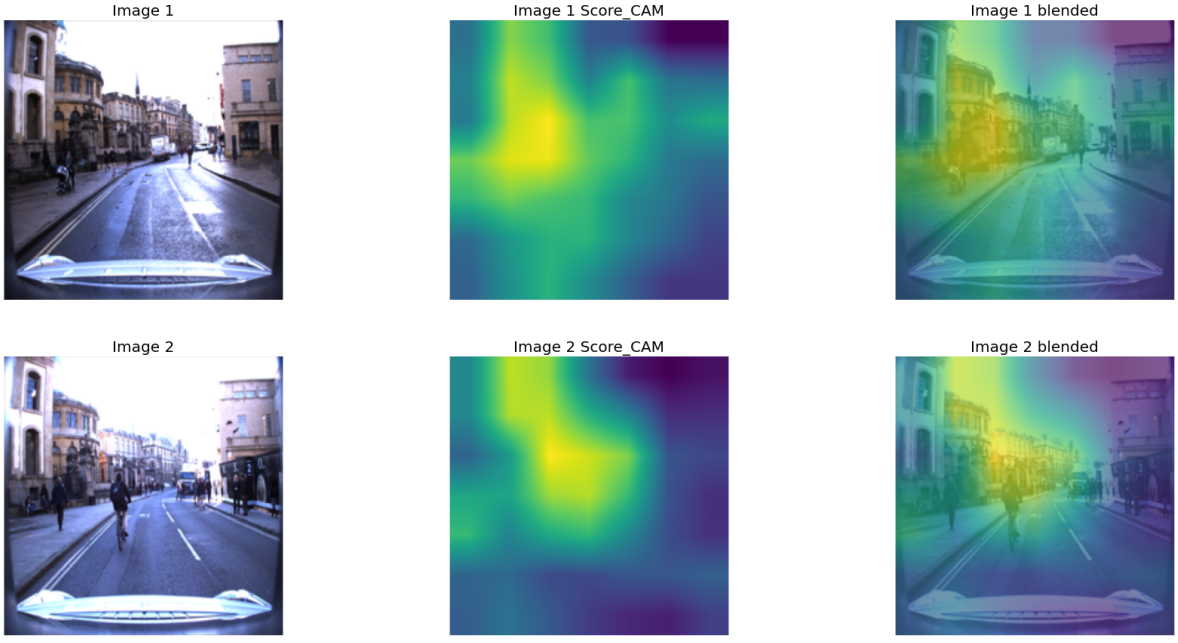
We decided to further evaluate the model's performance by generating and analyzing the Score-CAM images. Despite the relatively higher accuracy and AUC on the validation set, the score-CAM images reveal that the network does not focus on the right region of the input images all the time. For instance, in some cases, the network focuses more on cars, buildings, trees, or the sky instead of the road. A few of these Score-CAM images are presented below.



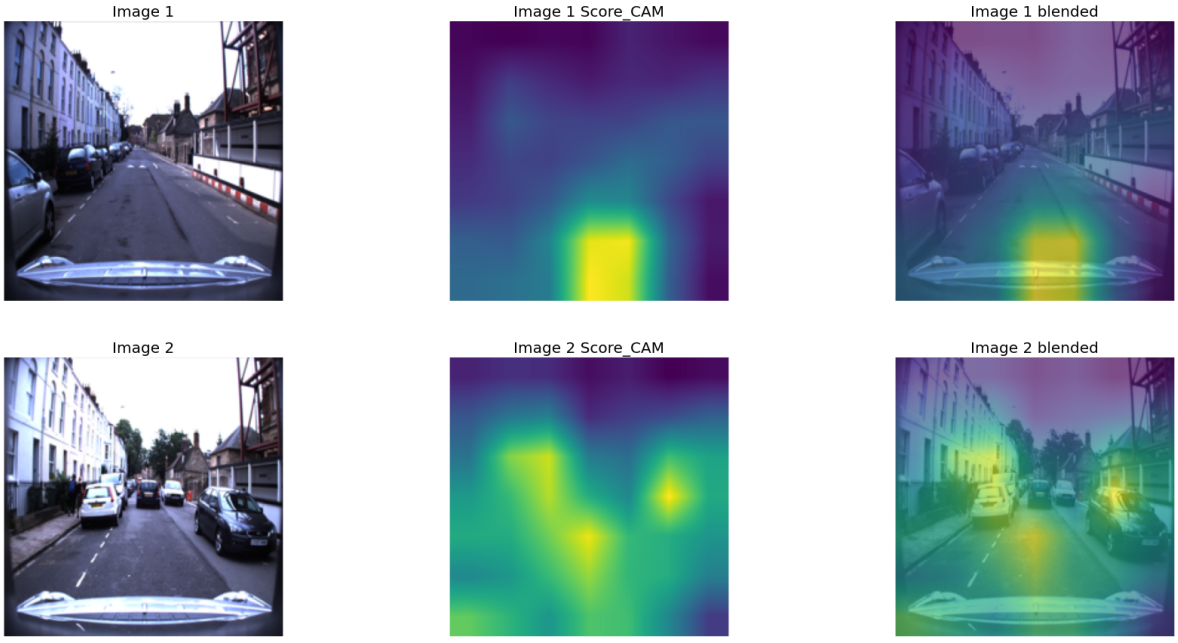
y_true:'no change'  y_pred:'no change'  P('no change'):0.63  P('change'):0.37



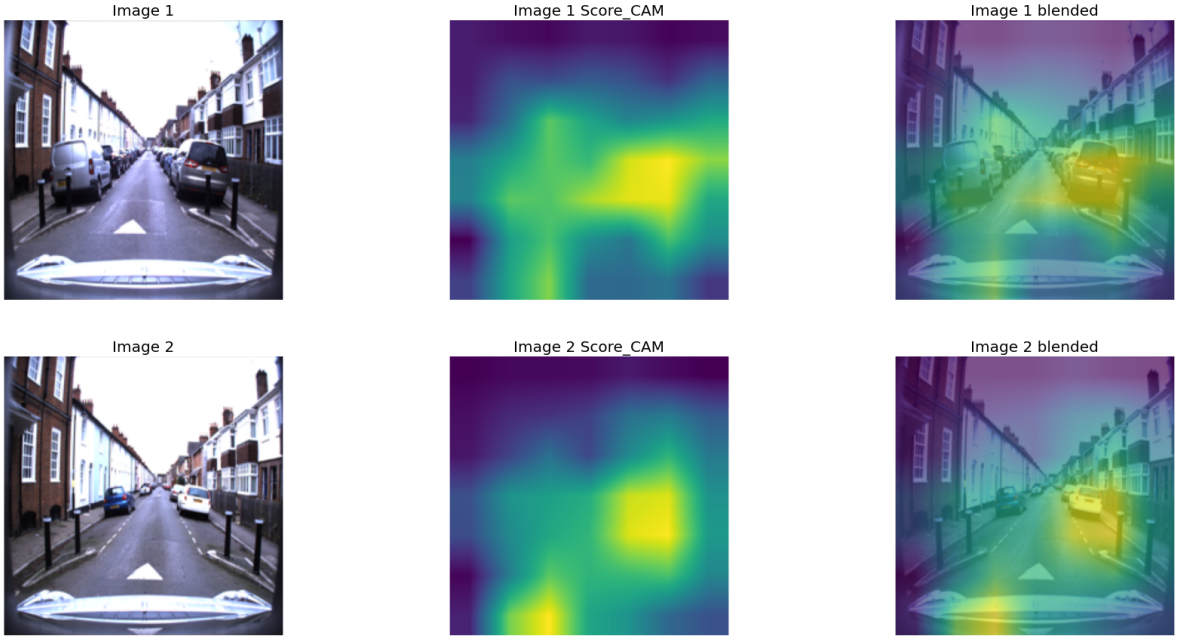

y_true:'no change'  y_pred:'no change'  P('no change'):0.7  P('change'):0.3



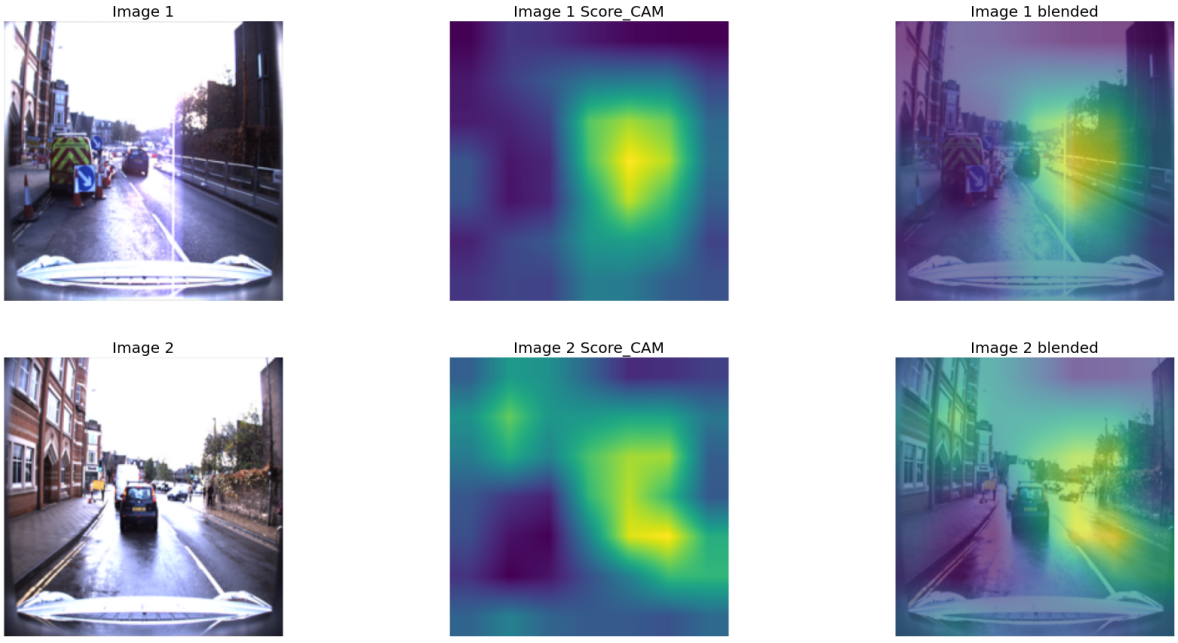
y_true:'no change'  y_pred:'change'  P('no change'):0.26  P('change'):0.74



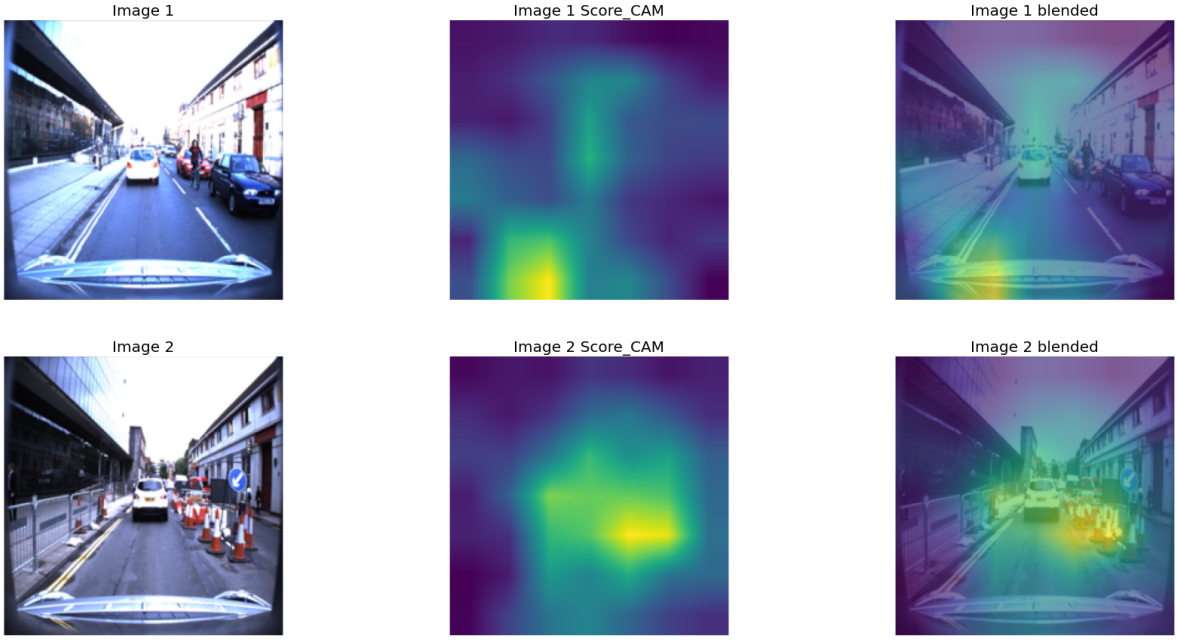

y_true:'no change'  y_pred:'change'  P('no change'):0.41  P('change'):0.59

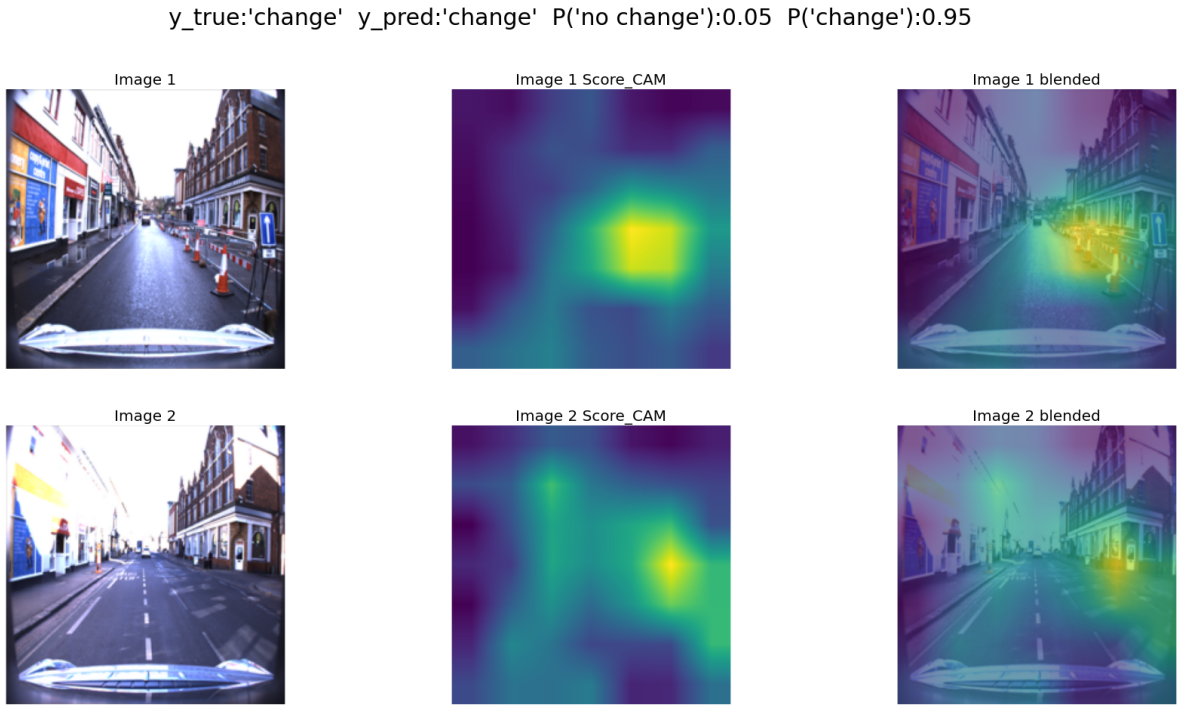


y_true:'change'  y_pred:'no change'  P('no change'):0.53  P('change'):0.47




y_true:'change'  y_pred:'no change'  P('no change'):0.59  P('change'):0.41






Furthermore, like in the previous experiment, a cross-validation experiment is also performed to further evaluate the generalization capacity of the model by training it with different data train/validation splits. Fifty-two validation folds are generated for this experiment using the approach that is discussed in the dataset chapter. For each validation fold, the pre-trained ResNet-50 model is fine-tuned for two epochs with the triplet instances. The triplet instances are split into the train and validation sets according to the validation fold root-segment ids. I.e., the triplet instances that their images belong to a root-segment which is split as the validation fold will be used as validation triplet instances. The feature-maps of all images in the LCDIP dataset are then extracted and stored by the fine-tuned ResNet-50 model. Finally, the second component is trained using the extracted feature-maps. The distribution of the validation AUC of the folds is visualized in figure 5.19. And the box plot of the model's validation AUC is illustrated in figure 5.20. As illustrated in figure 5.19, the neural network's AUC scores on the validation folds vary from 0.7 to 0.95. According to figure 5.20, roughly half of the AUC scores are in the range of 0.77 - 0.86. And the mean AUC score of all the validation folds is around 0.82.

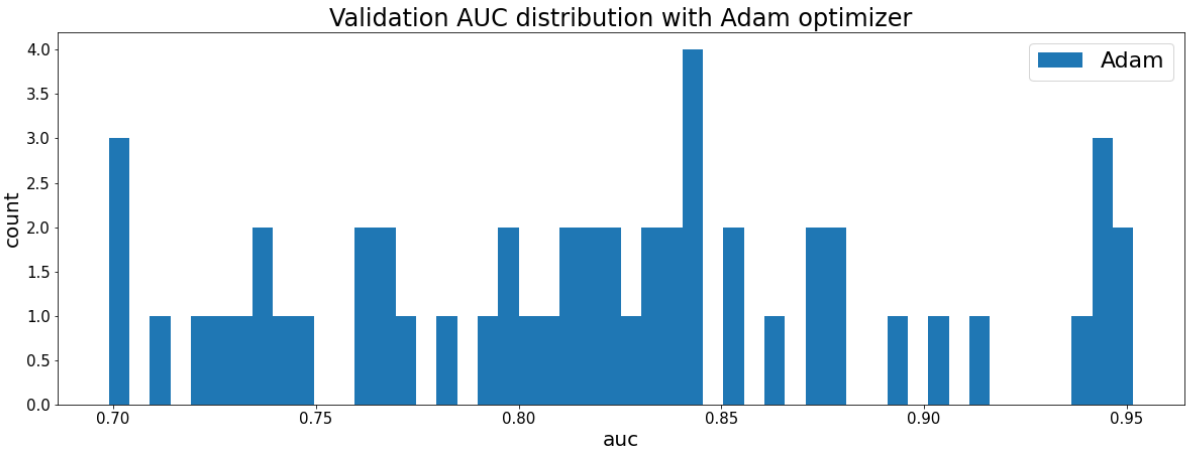



Figure 5.19. Validation AUC distribution of the cross-validation experiment with Adam optimizer.

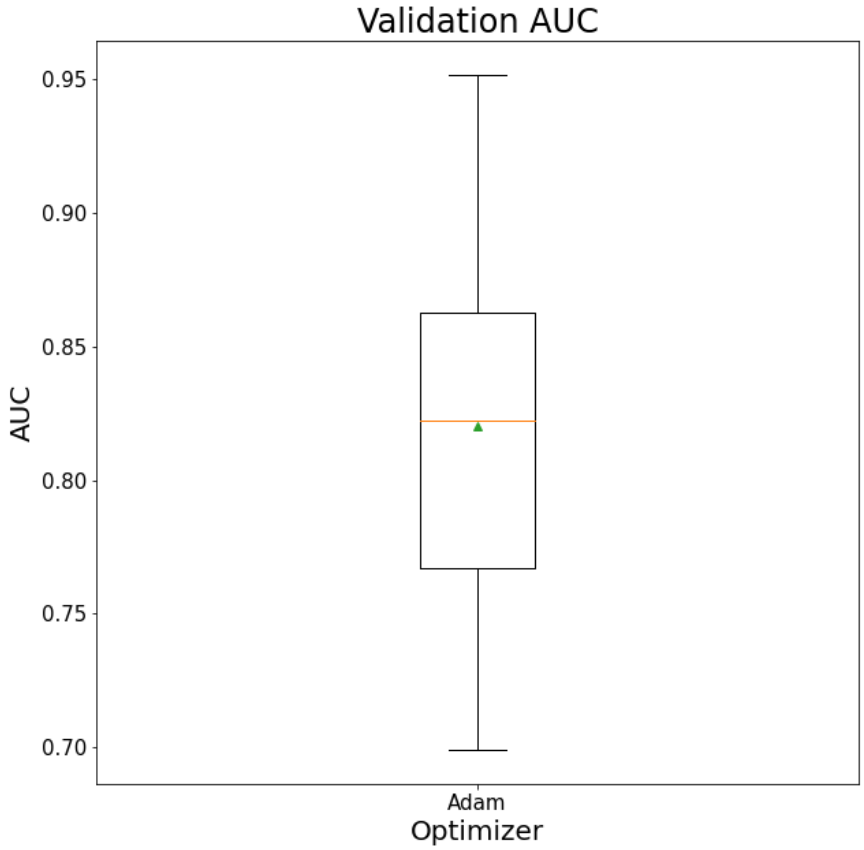



Figure 5.20. Validation AUC boxplot of the cross-validation experiment with the Adam optimizer.

According to the cross-validation experiments, it seems that fine-tuning the pre-trained ResNet-50 model with the Triplet Loss before training the second component of the proposed pipeline improved the final AUC score of the network. To prove it statistically, a T-test is performed. Here, we have cross-validation AUC scores of the network that uses the pre-trained ResNet model as the first component. These AUC scores will be referred to as the “pre-trained ResNet AUC scores”. The second collection is the cross-validation AUC scores of the network that uses the fine-tuned ResNet model by the Triplet Loss as the first component. These AUC scores will be referred to as the “fine-tuned ResNet AUC scores”. To run the T-test, the distribution of the AUC scores must have a normal distribution. The D’Agostino and Pearson’s test [36], [37] is performed to confirm the normality of the distribution of the pre-trained ResNet AUC score and the fine-tuned ResNet AUC scores. The null hypothesis in this test is that the collection has a normal distribution. And the alternative hypothesis is that the collection does not have a normal distribution. After running the test on the fine-tuned ResNet AUC scores, the p-value is 0.352. This means the p-value is large enough, and we cannot reject the null hypothesis. So, the fine-tuned ResNet AUC scores have a normal distribution. The p-value of the test on the pre-trained ResNet AUC scores is 1.83e-8. The low p-value means we can reject the null hypothesis, and the pre-trained ResNet AUC scores may not have a normal distribution. However, the box plot of figure 5.15 depicts that there are five outlier AUC scores under 0.59, whereas all other AUC scores are over 0.66. After removing those five AUC scores lower than 0.59, the p-value of the normal distribution test on the pre-trained ResNet AUC scores is 0.118. This p-value shows that by removing the outliers of the pre-trained ResNet AUC scores, the collection has a normal distribution, and it can be used in the T-test. Accordingly, the pre-trained ResNet AUC scores with removed outliers and the fine-tuned ResNet AUC scores both have normal distributions. After running the T-test on these two AUC score collections, the p-value is 0.0109. The p-value is less than 0.05, so we can reject the null hypothesis in this T-test, and this means two collections are statistically different. The mean AUC score of the pre-trained ResNet AUC scores is 0.7936, and the mean AUC score of the fine-tuned ResNet AUC scores is 0.8203. Since the T-test shows a statistical difference in the AUC score collections, we can say that the network that uses the fine-tuned ResNet-50 model with the Triplet Loss as the first component outperformed the network that uses the pre-trained ResNet-50 model by 13%.

# 6. Discussion

This work's main goal is to detect the changes in the streets' drivable area using camera images. As discussed in the introduction chapter, high definition maps (HD maps) provide detailed information about the autonomous vehicle's surrounding environment like roads, traffic lights, traffic signs, etc. The HD maps contain accurate information about the drivable areas of the streets which vehicles can go. Furthermore, the HD maps can be used to localize the autonomous vehicles because of the very accurate information of the surrounding permanent objects like buildings, traffic signs, etc. Therefore, HD maps can play a significant role in the future of autonomous vehicles. Since the details of these maps must be very accurate all the time, it is essential to update them right after any changes in the environment. However, detecting such changes could sometimes be challenging, especially if the difference is minimal. The changes in the streets' drivable area are critical changes that must be detected and applied to the HD maps as fast and accurately as possible. Since autonomous vehicles rely on HD maps to drive on the streets, any delay in detecting or updating the drivable area of streets in the HD maps may lead to severe threats and problems. Therefore, the goal of this work to automate the change detection in the drivable area of the roads is an essential step to keep the HD maps up to date.

## 6.1. Detecting Street's Drivable Area Long-Term Changes

As discussed in the related work chapter, some works try to detect the changes in the environment based on the camera images. These works generally try to detect any changes in the environment regardless of whether the change is temporary or permanent and whether the change affects the drivable area of the road or not. Therefore, we cannot directly use the existing works to detect the changes in the drivable area of roads for updating the HD maps. For example, a temporary object on the street may be correctly detected as a change based on the existing works, but it does not affect the street's drivable area since the change is temporary. As a result, the streets' drivable area change detection approach based on the camera images proposed in this thesis is a pioneering work.

## 6.2. Data Limitation

Since this work focuses on detecting the changes in the drivable areas of streets, it is necessary to use a dataset of images that show the roads at different times. So, the dataset must contain the street images before and after any changes in its drivable area. However, collecting these images needs a very long time and considerable effort since the changes in streets' drivable areas do not happen very often. As a result, collecting such images to create a dataset may take a few years. As discussed in the dataset chapter, the CDIP and LCDIP datasets used in this work are generated from the Oxford RobotCar dataset. The Oxford RobotCar dataset is a dataset that contains images of the streets taken by a camera mounted to a car gathered in 18 months. There are some changes in the roads covered by the Oxford RobotCar dataset images during the 18 months. However, the number of these changes is deficient, and it hinders the ability of the used machine learning approach to generalize the classification on unseen data.

## 6.3. Outcomes

As discussed before, this thesis is a pioneering work to detect the permanent or long-term changes in the street's drivable area. This is the first attempt to detect the changes in the roads' drivable area using the camera images. Moreover, as expressed in section 6.2, gathering data for this problem takes considerable time and effort. The dataset that is used for this work has a very limited number of changes in the drivable area of the roads. Despite all these limitations, this work could achieve a relatively good performance in detecting the changes. The best performance in experiment 5.1 achieved by the drivable area change detection model that uses the pre-trained ResNet-50 model as the first component to extract the feature-maps from input pair images. The extracted feature-maps are concatenated and fed into a multi-layer perceptron neural network with one hidden layer (model 1). In experiment 5.2, the model that uses concatenated 6-channel input images with a 6 to 3 channel convolutional neural layer had slightly better performance (model 2). The drivable area change detection model with pre-trained ResNet-50 model as the first component and a multi-layer perceptron neural network with two hidden layers as the second component is tested in experiment 5.3 (model 3). Experiment 5.4 tested a drivable area change detection model that used a fine-tuned ResNet-50 model using Triplet-Loss as the first component and a multi-layer perceptron neural network with two hidden layers as the second component (model 4). Figure 6.1 illustrates the validation accuracies of the trained models in different experiments. Except for model 2, the validation accuracies of

other models have been increased as more experiments are carried out. It appeared that concatenating input image pairs to a single image with 6 channels (RGBRGB) hindered the first component from extracting relative feature-maps. The highest validation accuracy is achieved by the model that used fine-tuned ResNet-50 model as the first component. Therefore, it seems that fine-tuning the pre-trained ResNet-50 model with Triplet-Loss improved the performance of the model in extracting the feature-maps. Considering the achievements of this work the better results can be achieved by investing more time in investigating new approaches and gathering more data to have a more extensive database in future works.

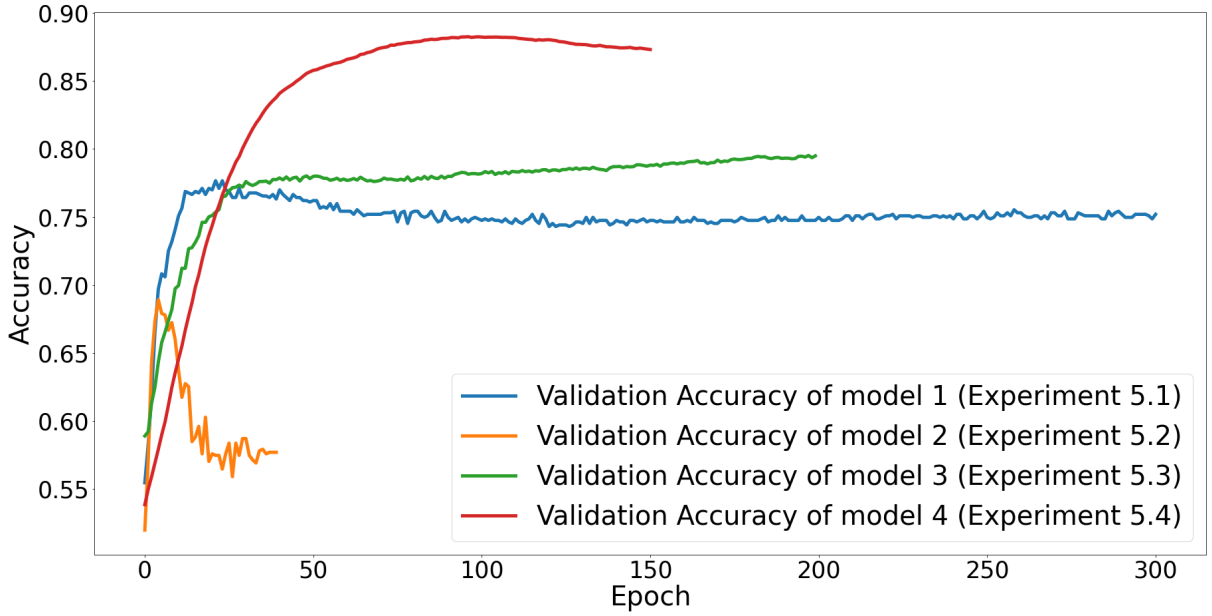



Figure 6.1. The drivable area change detection model of experiment 5.2 achieved the lowes accuracy among all experiments. The model of experiment 5.4 has the highest validation accuracy.

| | Validation Accuracy | Validation AUC |
|---|---|---|
| Model 1 (Experiment 5.1) | 77.67 % | - |
| Model 2 (Experiment 5.2) | 68.91 % | - |
| Model 3 (Experiment 5.3) | 79.52 % | 88.91 % |
| Model 4 (Experiment 5.4) | 88.26 % | 93.45 % |

Table 6.1. The highest validation accuracy achieved by the drivable area change detection model in experiment 5.1 is 77.67%. The validation accuracy of model 3 is 79.52%. This shows a 2.3% improvement in validation accuracy. The validation accuracy of model 4 shows an 11% improvement compared to the validation accuracy of model 3.

# 7. Conclusion

High Definition maps (HD maps) play a significant role in localizing and routing of the modern autonomous vehicles. HD maps provide detailed information about the streets, traffic signs, traffic lights, surrounding buildings, etc. Therefore, it is critical to keep the HD maps up to date at all times. This means any change in the surrounding environment must be detected and integrated into the HD maps as soon as possible. Automating the detection of such changes, therefore, will ease the effort of keeping the HD maps up to date. Images of the streets are used as the primary source to detect the changes in the streets' drivable areas in this work. The images are taken on different days by a camera mounted on top of a vehicle. A Streets' drivable area change detection pipeline is proposed to detect the changes that have to be applied to the HD maps. The proposed pipeline consists of two artificial neural network components. The first component extracts feature-maps from two input images that show a particular section of a street on two different days. The second component, then, is responsible for classifying the feature-maps of the input pair images whether there is any change in the drivable area of the street or not. Despite being a pioneering work to detect the changes in the drivable areas of the roads based on the camera images and the difficulty of gathering street images that show a change in the drivable area of the road, promising results are obtained.

As future works, a larger dataset must be gathered. The CDIP and LCDIP datasets are used in this work for the training of the drivable area change detection model. However, these datasets have a limited number of positive instances. I.e. datasets do not have enough image pairs that show a change in the drivable area of the streets. So, a higher performance can be achieved by gathering more data. Moreover, semantic segmentation approaches can be used to narrow down the focus of the model on the streets instead of the other parts of the image like buildings.

# 8. Acknowledgements

This work is funded by European Social Fund via Smart Specialization project with Bolt. The computational environment and resources for this work is provided by the High Performance Computing (HPC) center of the Institute of Computer Science at the University of Tartu. I would like to thank my supervisors, Dmytro Fishman and Dr. Naveed Muhammad, for their outstanding support and inspiring ideas throughout the project.

# References

[1] Tampuu, Ardi, et al. "A survey of end-to-end driving: Architectures and training methods." IEEE Transactions on Neural Networks and Learning Systems (2020).

[2] Mariani, Stefano, Giacomo Cabri, and Franco Zambonelli. "Coordination of autonomous vehicles: taxonomy and survey." ACM Computing Surveys (CSUR) 54.1 (2021): 1-33.

[3] SAE International. 2018. Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles.

[4] Menon, Nikhil, et al. "Shared autonomous vehicles and their potential impacts on household vehicle ownership: An exploratory empirical assessment." International Journal of Sustainable Transportation 13.2 (2019): 111-122.

[5] Yurtsever, Ekim, et al. "A survey of autonomous driving: Common practices and emerging technologies." IEEE Access 8 (2020): 58443-58469.

[6] Seif, Heiko G., and Xiaolong Hu. "Autonomous driving in the iCity—HD maps as a key challenge of the automotive industry." Engineering 2.2 (2016): 159-162.

[7] Vardhan, Harsha. "HD Maps: New age maps powering autonomous vehicles." Geospatial world (2017).

[8] Alcantarilla, Pablo F., et al. "Street-view change detection with deconvolutional networks." *Autonomous Robots* 42.7 (2018): 1301-1322.

[9] Varghese, Ashley, et al. "ChangeNet: A deep learning architecture for visual change detection." *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*. 2018.

[10] Fediukov, Vladyslav. "Detection of changes in maps using LiDAR point clouds."

[11] W. Maddern, G. Pascoe, C. Linegar and P. Newman, "1 Year, 1000km: The Oxford RobotCar Dataset", *The International Journal of Robotics Research (IJRR)*, 2016.

[12] W. Maddern, G. Pascoe, M. Gadd, D. Barnes, B. Yeomans, and P. Newman, "Real-time Kinematic Ground Truth for the Oxford RobotCar Dataset", in *arXiv preprint arXiv: 2002.10152*, 2020.

[13] López, Victoria, et al. "An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics." *Information sciences* 250 (2013): 113-141.

[14] Wang, Haofan, et al. "Score-CAM: Score-weighted visual explanations for convolutional neural networks." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. 2020.

[15] https://github.com/haofanwang/Score-CAM

[16] Zou J., Han Y., So SS. (2008) Overview of Artificial Neural Networks. In: Livingstone D.J. (eds) Artificial Neural Networks. Methods in Molecular Biology™, vol 458. Humana Press. https://doi.org/10.1007/978-1-60327-101-1_2

[17] G. P. Zhang, "Neural networks for classification: a survey," in IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), vol. 30, no. 4, pp. 451-462, Nov. 2000, doi: 10.1109/5326.897072.

[18] G.-S. Kalanit and M. Rafael, The human visual cortex, Annual Review of Neuroscience, 27 (2004), 649–677.

[19] Qin, Zhuwei, et al. "How convolutional neural network see the world-A survey of convolutional neural network visualization methods." arXiv preprint arXiv:1804.11191 (2018).

[20] K. He, X. Zhang, S. Ren and J. Sun, Deep residual learning for image recognition, in Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, 2016, 770–778.

[21] K. Simonyan and A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint, arXiv:1409.1556.

[22] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke and A. Rabinovich, Going deeper with convolutions, in Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, 2015, 1–9.

[23] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot et al., Mastering the game of go with deep neural networks and tree search, Nature, 529 (2016), 484–489.

[24] S. Hochreiter. Untersuchungen zu dynamischen neuronalen netzen.
Diploma thesis, TU Munich, 1991.

[25] Bradley, David M. Learning in modular systems. CARNEGIE-MELLON UNIV PITTSBURGH PA ROBOTICS INST, 2010.

[26] Glorot, Xavier, and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks." Proceedings of the thirteenth international conference on artificial intelligence and statistics. JMLR Workshop and Conference Proceedings, 2010.

[27] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. arXiv:1409.0575, 2014.

[28] Ramchoun, Hassan, et al. "Multilayer Perceptron: Architecture Optimization and Training." IJIMAI 4.1 (2016): 26-30.

[29] Ringnér, Markus. "What is principal component analysis?." Nature biotechnology 26.3 (2008): 303-304.

[30] Shwartz-Ziv, Ravid, and Naftali Tishby. "Opening the black box of deep neural networks via information." arXiv preprint arXiv:1703.00810 (2017).

[31] Alain, Guillaume, and Yoshua Bengio. "Understanding intermediate layers using linear classifier probes." arXiv preprint arXiv:1610.01644 (2016).

[32] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014.

[33] M. T. Ribeiro, S. Singh, and C. Guestrin. Why should I trust you?: Explaining the predictions of any classifier. In Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, pages 11351144. ACM, 2016.

[34] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Learning deep features for discriminative localization. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 2921–2929, 2016.

[35] O'Shea, Keiron, and Ryan Nash. "An introduction to convolutional neural networks." arXiv preprint arXiv:1511.08458 (2015).

[36] D’Agostino, R. B. (1971), “An omnibus test of normality for moderate and large sample size”, Biometrika, 58, 341-348

[37] D’Agostino, R. and Pearson, E. S. (1973), “Tests for departure from normality”, Biometrika, 60, 613-622

[38] Mueller, J., Thyagarajan, A.: Siamese recurrent architectures for learning sentence similarity. In: AAAI. (2016) 2786–2792

[39] Koch, G.: Siamese neural networks for one-shot image recognition. Master’s thesis, University of Toronto, Canada (2015)

# Appendix

## I. Glossary

LiDAR - Light Radar
GNSS - Global Navigation Satellite System
GPS - Global Positioning System
INS - Inertial Navigation System
HD map - High Definition map
SLAM - Simultaneous Localization and Mapping
CNN - Convolutional Neural Network
FCN - Fully Convolutional Network
MLP - Multilayer Perceptron
ResNet - Residual Neural Network
ADAM - Adaptive Moment Estimation
SGD - Stochastic Gradient Descent
PCA - Principal Component Analysis
AUC - Area Under the ROC Curve
ROC curve - Receiver operating characteristics curve

# II. License

Non-exclusive licence to reproduce thesis and make thesis public

I, Navid Bamdad Roshan,
( author's name)

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

   Change Detection in HD-Maps Using Camera Images for Autonomous Driving,
   ( title of thesis)

   supervised by Dmytro Fishman and Naveed Muhammad.
   (supervisors' name)

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' in-tellectual property rights or rights arising from the personal data protection legisla-tion.

Navid Bamdad Roshan
07/05/2021