UNIVERSITY OF HERTFORDSHIRE

Department of Computer Science

MSc Artificial Intelligence And Robotics

7COM1039-0109-2019 and Advanced Computer Science Masters Project

10-09-2020

Thermal Image Recognition

Name: **Abdul Basit**
Student ID: **17062153**
Supervisor: **Dr Na Heilan**

Abstract:

Face image recognition is in demand and now days many researchers are holding research to enhance the technique to achieve better accuracy. It's also relevant and useful to the security issues so both commercial and military fields have their interest in it. Visible face image recognition achieved almost very good accuracy but it fails or struggles in low light or when the person's face is covered. For thermal images recognition, thermal image is required to train the model but due to less images of thermal, it's hard to train the model. The objective of the project is that by using different types of filter, the accuracy of the thermal image recognition can be better. And impact of the data size on the thermal image recognition accuracy. The project has proven that by using different types of filters in pre-processing of the data, the thermal image recognition accuracy can be improved. Also, it compared the result with different type of filters which were used in the pre-processing of the data. The results suggest that gaussian filter gives better results as compare to other filters.

Acknowledgment

In the Name of Allah, the Most Gracious and Most Merciful

Firstly, I would like to express my special gratitude towards my supervisor, Dr. Na Helian, for providing me an opportunity to work on this project and her untiring help and feedback throughout the project. Secondly, I would like to thank my family and specially to my mother and late father for supporting me to fulfil my dream.

Table of Contents

List of Figures

List of Tables

1. Introduction

Nowadays, most popular research domain in both the military and the commercial area is the face recognition. Most of the face recognition techniques so far used the visible images to recognize people. In recent year, many of the techniques provide very good accuracy when they recognize the visible images. These techniques work well however, when there is proper light or the images are clear, but in low light or night-time, these techniques or the model struggle to recognize the face.

Most of the crimes appear at night. Often robbery and trespass happen during night-time and even if one has cameras installed to recognise faces, they mostly fail due to low light, night time conditions since they are designed to recognise face features only with a clear-visible image.

However, there is now more data set of images, especially for the face image visibility. Moreover, after the deep learning evaluation, things happen more quickly as compared to the past. Now, the machine can learn more advanced tasks which were easy for the humans but for the machine, they were comparatively hard to do since the machine can understand only numbers.

Deep learning has a hunger for the data. The more data one has, the more accuracy of the model they might be able to achieve. That's why more accuracy of face recognition is achieved because the amount for the data set of the visible face is very vast but the amount for the thermal images of the faces are less as compared to the visible face images. There is small amount of data sets with the limited amount of the images of the subjects. It's very difficult when it comes to recognizing the thermal face while using the deep learning model for recognition.

So to overcome this situation, we can use the thermal image for the cross domain face recognition. Since the thermal images are low in the resolution, it's hard to do the cross-domain recognition and this increase the domain gap between thermal image recognition and visible face recognition.

Also, many of them did not apply image filters to the images. They did not focus on the step of the pre-processing with different time of the filters and this is the gap which the researcher has addressed in this project. In the following project, after having a careful read, one can see what results are achieved after applying different types of filter. Does it benefit to apply filters? If it does, then to what extent it helps to the model for achieving the accuracy.

1.1. Aim of the Project

The aim of the project is that by applying different kind of filters in the step of data pre-processing, the accuracy of the deep auto encoder decoder model can be enhanced and also to compare the different set of data size which shows that the more data one has, the more it helps to enhance the accuracy of the model.

In this project, the researcher implements the deep auto encoder model for the thermal image recognition and then uses different filters in the pre-processing step of the deep auto encoder

decoder model. The research also shows that which filter performs best among them to achieve higher accuracy of the model. The filters applied by the researcher in this project are: Mean, Median, Gaussian.

To prove that applying filters in the data pre-processing step will help to enhance the accuracy of the thermal image recognition, experiments and results are discussed in the experiment and evaluation section.

2. Literature review:

(K. Mallat, 2019) , proposed new solution by using a cascaded refinement network. The method generates visible like images without using a large amount of data set. By using a contextual loss function, the proposed network is inherently scaled and rotation invariant. After generating the visible like an image from the network they compare those images with the visible images from the gallery using deep learning models. For generating images they use CRN method cascaded refinement network.

The researcher uses VIS-TH face database which consists of 2100 images in a total of 50 different subjects. For face recognition, he used OPEN Face Network and light CNN. The result shows that there was an improvement of 35 percent from 16 % to 22%.

(Samah A.F.Mansoor, 2019), researcher proposed, approach is that they make build a deep convolutional neural network composed of 23 layers named TRI NET. The architecture of the network is convolution followed by Max pooling repeat it for one more time followed by flattened and in last layer, fully connected feed-forward network. The system is composed of a Thermal Camera PC and a set of visible and thermal face images. The researcher uses the DUFO database which is composed of 3400 images and total subjects are 13. The researcher pre-processed the images by image normalization which removes the dead pixel and in second step, local variation reduction of images, this adjusts the local variations. On the DHUFO database, they get 98.50% result and the training time is between 10 to 20 hours.

(X. Dong, 2019), proposed a secure visual fused face recognition system using a non-linear hashing technique. For extracting feature, the researcher used a deep neural network model pre-trained by the visible images named Insight Face. Non-linear hashing provides an extra layer of protection in the matching for face recognition. It was also emphasized that the pre-trained model on visible images are useful for the feature extraction for the thermal images. Researcher uses EUROCOM VISTH face date set for the experiments. The total number of images is 2100 for this data set.

(Lin, 2019), the researcher proposed a method called model fusion. This model fusion has three main approaches these are linear support vector machines classifies, convolutional neural network, and ordinary least square. The researcher used a grid of 22 thermal face points base on physiological information extracted for training linear SVC. UCH Thermal temporal face and pucv Thermal face databases are used in this method.

For the pre-processing, the pixel value of thermal images is normalized between 0 to 1 and the images are normalized to a uniform size. The CNN model is VGGNET and CNN is composed of four convolutional layers and two fully connected layers. The proposed approach gives 98.96 percent accuracy on normal thermal images.

(Ekenel, 2019), uses the deep autoencoder method to create a thermal image as an output and then matches that output with the gallery. The researcher uses U-Net shape architecture for deploying autoencoder and total trainable parameters are 14789059. The researcher has done the pre-processing on both visible and thermal images. First, the researcher applies mean filter followed by the difference of Gaussian filter, and has performed alignment to the data set. Researcher uses three different data set which are CARL data Set UND-XI Data set and EUROCOM Data set. Performance increases by 2 % by using the alignment.

2.1. Summary:

Different algorithms using different data set, environment and their results are discussed in this section .Also, shown below are some research gaps of these papers. Consider the following table, which shows the general summary of the research papers reviewed.

Reviewed papers	Overview	Data set	Research gap	results
(K. Mallat, 2019)	They generate visible like image using CRN method and then compare with the gallery using Light CNN and OPEN Face network	VIS-TH face database. total images of 2100	They did not account on pre-processing of the data set and for deep learning, there should be more data for better result	They reported a 22 percent result
(Samah A.F.Mansoor, 2019)	They use CNN with 23 layers called it TRI NET	DUHFO data set contain 3400 images in total	Its time is taken and costly as well. specific on one data set	Accuracy on DHUFO database is 98.5 percent
(X. Dong, 2019)	They use the pre-trained model of visible images named as Insight Face	VIS-TH face database. total images of 2100	They did not train images on a new model and also does not account for pre-processing	They reported result about fusion score level
(Lin, 2019)	Their fusion model is composed of three things. SVM classifies, CNN and OLS	UCH Thermal temporal face and pucv Thermal face data set	They did not account about the pre-processing of the data	They reported 98.96 % accuracy on normal thermal images

(Ekenel, 2019)	U-Net architecture using deep autoencoder	CARL data Set ,UND-XI Data set and EUROCOM Data set	They did not take account for the various appearance	It increase by 2 % when they aligned that.

Table 1 Summary of Literature review

3. Methodologies:

The methodology is based on the experiments but to do experiments the deep learning model named as deep auto encoder decoder should be implemented so furthers experiments can be conducted. Researcher going to implement the deep auto encoder model which same as described by *Ekenel*, 2019. The author uses the segmentation model as a base and then modify that model in to deep auto encoder decoder for thermal image recognition. The total trainable parameters of *Ekenel*, 2019 architecture are 14,789,059 and the loss function of the network is mean square loss.

For the methodology purpose, researcher has followed (Ekenel, 2019) for the architecture, data set, and for the trainable parameters.

(Ekenel, 2019), uses U-Net architecture as a base and modify that to the deep autoencoder for the purpose. The model is comprised of (convolution + batch norm + relu)*2 followed by the max pooling and up convolution. The total trainable parameters of (Ekenel, 2019) architecture are 14,789,059 and the loss function of the network is mean square loss. The network takes input as the visible images and the output is thermal images. The figure below illustrates the architecture of (Ekenel, 2019).

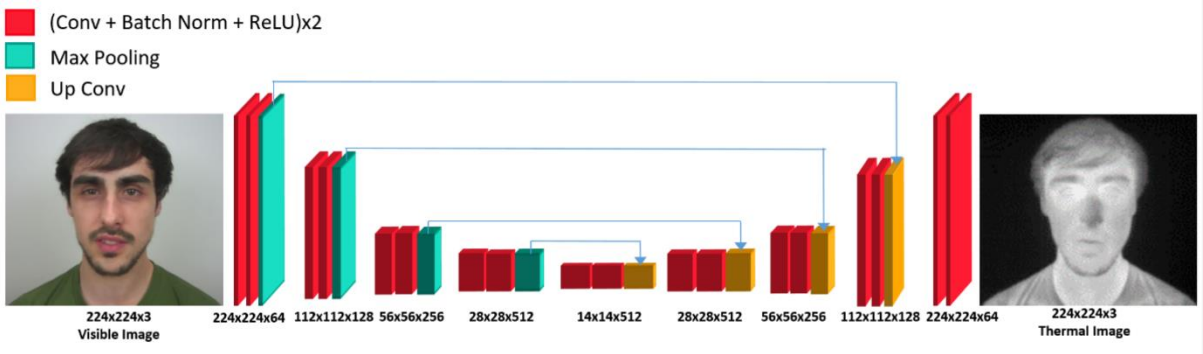

Figure 1 Architecture of Autoencoder

3.1. Data Set

For this project, Researcher is has used Carl data set. It contains 41 different subjects with NIR and LWIR visible paired images. Researcher has only used LWIR and visible images and in total, they are 4920 images. Half of them are visible and half of them are thermal.

3.2. Pre-processing:

In the pre-processing of the data for this project, different steps are involved, which are resizing the images and then application of different types of filters to the data set.

3.2.1. Resizing:

The images of the data set are resized to 224* 224 * 3 because the researcher *Ekenel*, 2019, uses this shape for the input of the model.

3.2.2. Mean Filter:

Mean filter replaces each pixel value in an image with the average value of its neighbours. It is usually considered as convolutional filter. The mean filter is used to blur the image. By blurring the image, it will remove the noise from the images. It also smoothens out the edges of the image. It can be implemented by using the function in OPEN CV library.

3.2.3. Median Filter:

It calculates the median of the pixel intensities that surround the centre pixel, consequently; it replaces the pixel intensity of the centre pixel. It removes the salt and paper noise of the image. It preserves the edges of an image but does not deal with the speckle noise of an image. It can be used by applying the median blur function in OPEN CV library.

3.2.4. Gaussian filter:

Gaussian filter is similar to mean filter but it also weighted the average of the surroundings pixel and has parameter sigma. Same as mean filter, it also blurs the images but it does a better job in preserving edges of the images. It can be run by using the Gaussian blur function in OPEN CV library.

3.3. Model Evaluate

The evaluation of the model will be on the value of the training loss and difference between training loss and validation loss. The less we have the value of training loss, and the difference between training loss and validation loss, the more accurate model will it be.

3.4. U- Net

U net is the model that is used for image segmentation and it became well known in 2015. It performed very well in the bioimage segmentation. The convolutional neural network gave the good result of the simple image but struggled for the complex ones. It was first designed for the medical image segmentation and after that, it was used for other problems as well.

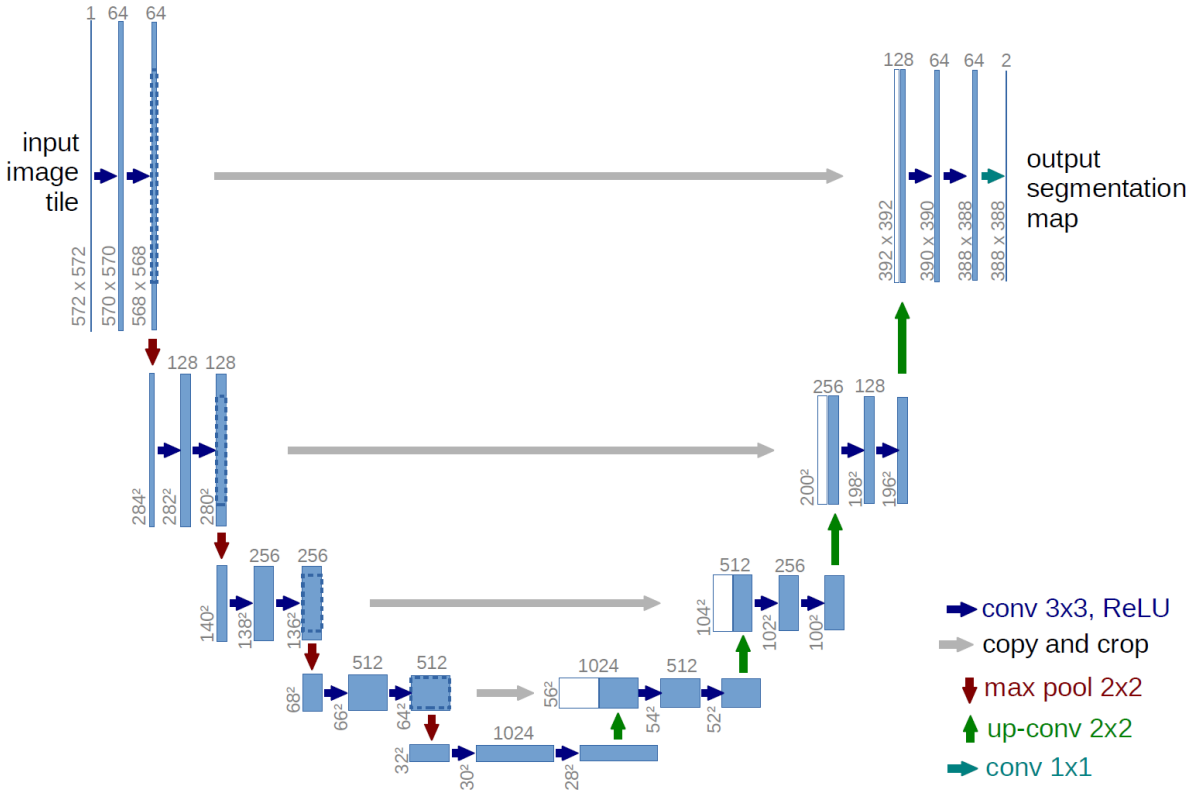

Figure 2 Architecture of U-Net

The architecture is composed of three main parts: the contraction, the bottleneck, and the expansion section. The contraction part is down sampling and it is made up of many contraction parts. Each contraction part consists of 3 into 3 convolutional layers followed by 2 into 2 max pooling.

The bottleneck layer is just a connected part between contraction and expansion; it is the same as the contraction part. The expansion part also consists of many expansion blocks and each expansion block had 3 *3 convolution layer followed by the 2 *2 max pool layer. The number of the expansion layer is the same as the number of the contraction layers so each feature could be a map on the output.

3.5. Transfer learning

For the implementation purpose either researcher has to develop U-Net segmentation model from the scratch and then modify that model in the form of deep autoencoder decoders or researcher can benefit from some other work which has a segmentation model in order for the researcher to modify that model according to his need and requirements.

3.6. Segmentation Model Repository:

There is an online repository of segmentations model. Yakubovskiy, 2019 provides U shaped segmentation model with various types of backbones for the segmentation model. Some of the types are as follow: VGG, ResNet, SE-ResNet, ResNext, SE-ResNetx, SENET154, DenseNet, Inception, MobileNet and EfficientNet. (Yakubovskiy, 2019)all backbones have weights trained on 2012 ILSVRC ImageNet data base.

4. Implementation:

4.1. Choosing the right backbone:

There are many types of backbones available, however, the researcher has to look and apply the right type. Therefore; on the basis of trainable parameters, the researcher will choose the backbone because (Ekenel, 2019); 2019, has 14,789,059. So researcher has to find the exact or the closet parameters for this project. At first, researcher tries with the resnet 34 but its trainable parameters were very high that was 24,438,804. Then, the researcher tries mobile net v2 and sees the parameters of it. It was not enough, or very less from the desired ones; it was 8011345. Resnet 18 has the closet trainable parameters which are 14,330,550. These are close to the parameters of (Ekenel, 2019).

4.2. Changing Segmentation model to auto encoder decoder

The main difference between U-net segmentation models and deep auto encoder decoder is that in the segmentation model there is last layer of sigmoid function and fully connected layer. If we remove these two last layers, the U net segmentation model can be modified to the deep auto encoder decoders. The model.layers.pop() command will remove the layer of the model. Following is the screenshot of code, explaining how the segmentation model is changed to deep auto encoders decoders.

```
BACKBONE = 'mobilenetv2'

model = sm.Unet(BACKBONE, input_shape = (224, 224, 3))

model.layers.pop()
model.layers.pop()

x = tf.keras.layers.Conv2D(3, (1,1))(model.layers[-3].output)
model = tf.keras.models.Model(model.layers[0].output, x)
```

Figure 3 Changing segmentation model to auto encoder decoder

4.3. Resources issues

In the development to run the model, the main problem is computational power. To process those images it requires a high level of the graphic source so the computational can be done. As stated by (Ekenel, 2019), it took about average 5 hours to compute the model on tesla k80 GPU. While on researcher's laptop, there is a limited amount of computational access. It may take ages on researcehr's laptop to train the model because when the researcher tried to run that, it took almost 8 hours to run about 32 Epoch. And the researcher needs to run the model on 200 EPOCH.

4.4. Google Colab

To resolve the computational issues, the researcher had to use the google colab to access the computational power online. It provides online access to tesla k80 so one can run their model on the virtual machine provided by the colab. But in order to do it, the researcher had to

integrate his code with the google colab and installed the specific libraries for the project so it could be worked out.

4.4.1. Integration of Colab

To use the google colab, first we need to upload the data set to the google drive to use that, after that one needs to mount the drive with the colab soone can have access to the data set. One will write down the mount code, and when it runs, it gives a URL. On that URL; there is a password generated. After copying the password you drive fill to be mounted. The following screenshot explains how the researcher mounted the drive with google colab

```
[ ]  from google.colab import drive
     drive.mount('/content/gdrive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2

Enter your authorization code:
··········
Mounted at /content/gdrive
```

Figure 4 Integeration of Colab

After mounting your drive the next step is to give a path of your data set so you can import that for your project. I am using the glob variable for the import of the data set. The following screenshot is how I access my data and import that for my project.

```
[ ]  classic = glob("/content/gdrive/My Drive/ThermalImageTest/Database1/*/*/classic")
     thermal = glob("/content/gdrive/My Drive/ThermalImageTest/Database1/*/*/thermal_bmp")
     X = list()
     Y = list()

     for subj in range(len(classic)):
         X.extend(sorted(glob(classic[subj] + '/*.bmp')))
         Y.extend(sorted(glob(thermal[subj] + '/*.bmp')))
```

```
[ ]  sample = np.random.randint(0, len(X))
     X[sample], Y[sample]
```

```
('/content/gdrive/My Drive/ThermalImageTest/Database1/29/session_2/classic/29S2CAR5.bmp',
 '/content/gdrive/My Drive/ThermalImageTest/Database1/29/session_2/thermal_bmp/29S2BAR5.bmp')
```

Figure 5 Integration of Data Base

As it can be seen in the screenshot, google drive where I had uploaded my data in my google drive and this code import the data in the pair. Same subject image of the classic and thermal image of that subject.

4.4.2. Google Colab Resource Limit:

The google colab resources are not guaranteed and are limited to the user. The maximum time for one session is up to 12 hours which is also not guaranteed. If someone's model training requires more than 12 hours of training on google colab, then it will be not entertained and somehow if during their training, they are disconnected or some other issue occurs, this will result in data loss.

Apart for the time notebook can be an idle maximum for 90 minutes. One can leave their notebook for 90 minutes. If it is more than 90 minutes, then the connection will be lost and the work will also be wasted.

4.4.3. Colab Pro:

To overcome the problem with the google colab, there was an option of colab pro which hs more computational power and the limit to use that increases from 12 hours to 24 hours. However, the google colab pro is only available in The United States of America. It's not available outside USA.

As one can see in the screenshot below:

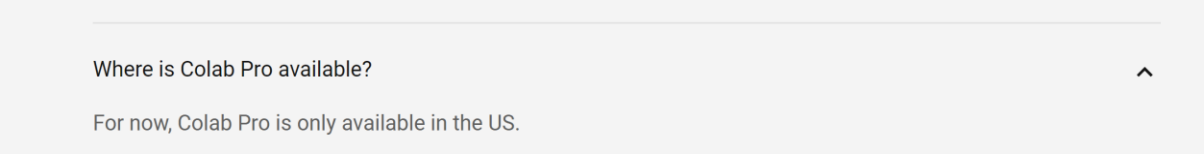

Figure 6 Colab Pro

4.4.4. Storage issues

While using google colab, one will encounter storage issues of the drive. When the runs the virtual machine takes the space from the user's google drive and that's why there will be storage issues.

4.4.5. Upgrading storage capacity

To use the google colab, you need to but the google drive subscription for the extra storage. if you go with the 100 GB storage it will cost £ 1.5 per month.

4.5. Changing of back bone:

Due to limitation of the resources, the researcher had to reduce the trainable parameters. The researcher changed the backbone of the architecture from Resnet 18 to mobile net v2 because the total trainable parameters of the mobile net v2 are 8011251 which are less from the resnet18 architecture.

4.6. Mobile net v2

Mobilenetv2 is used for visual recognition including classification object detection and semantic segmentation. It used depth-wise separable convolution as efficient building blocks. It has linear bottlenecks between the layers and shortcut connection between the bottleneck .The trainable parameters is less if compared to resnet18. It's a light weight model that's why researcher is using this model. Following is the basic structure of mobilenetv2.

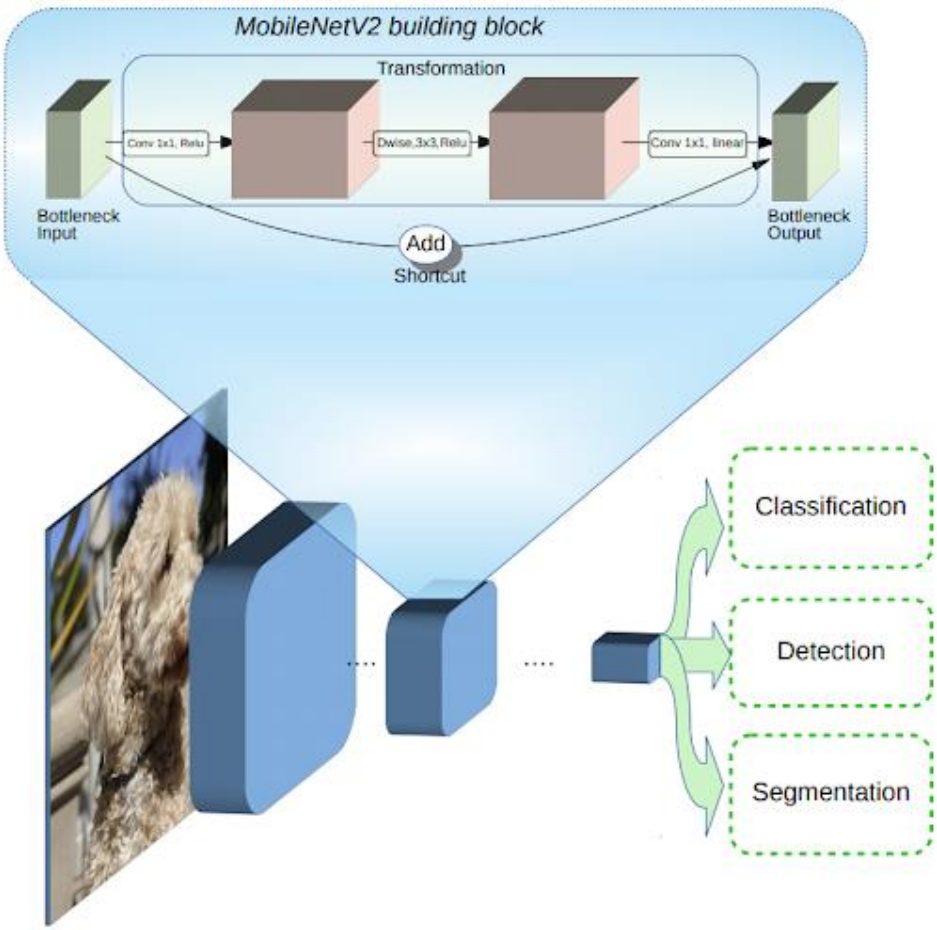

Figure 7 Architecture of Mobile Net V2

5. Experiments Design and Results

The aim of the experiments design and result is to show the data size impact and filter type impact on the performance of the deep auto encoder decoder model. To execute this, researcher had designed the experiments and formed the set of sizes for the data set and implemented the filter in the step of pre-processing of the data. Following are the details about the data sizes and the implementation of the filters:

5.1. Data set 1

The total amount of images used in this data is 960 images. Half of them are classic images that are 480 in total, and half of them are thermal images which are 480 in total.

The ratio of training data is 80 percent from the whole data set which are 768 images, and 20% of the data is validation data or test data which is about 192 in total.

5.1.1. Data set 2

The total amount of images we have used in this experiment is 1200 images. Half of them are classic images that are 600 in total, and half of them are thermal images which are 600 in total.

The ratio of training data is 80 percent from the whole data set which are 960 images, and 20% of the data is validation data or test data which is about 240 in total.

5.1.2. Data set 3

The total amount of images used for this experiment is 1560. Half of them are classic images: 780 in total and half of them are thermal images which are 780 in total.

The ratio of train data is 80 percent from the whole data set which are 1248 images and 20 percent of the data is validation data or test data which is about 312 in total.

5.2. Types of filters

The filter which I am going to use for the pre-processing of the data is as follows Mean filter, Median Filter, Gaussian Filter

5.2.1. Filter 1(Mean filter):

The mean filter is used to blur the image. By blurring the image it will remove the noise from the images it also smooth the edges of the image. It can be implemented by using blur function in OPEN CV library. Following is the screenshot of the code which is implemented by the researcher.

```
#Apllying Mean filter
meanf_x = cv2.blur(resized_x,(5,5))
meanf_y = cv2.blur(resized_y,(5,5))
```

Figure 8 Applying Mean Filter

5.2.2. Filter 2(Median Filter):

It calculates the median of the pixel intensities that surround the center pixel then it replaces the pixel intensity of the centre pixel. it removes the salt and paper noise of the image. It preserves the edges of an image but n does not deal with the speckle noise of an image. It can be used by using median blur function in OPEN CV library.

Below is the screenshot to show how it is implemented in the code:

```
#apply median filter
meanf_x = cv2.medianBlur(resized_x, 21)
meanf_y = cv2.medianBlur(resized_y, 21)
#apply guassian filter
```

Figure 9 Applying Median Filter

5.2.3. Filter 3(Gaussian filter):

Gaussian filter is similar to mean filter but it also weighted the average of the surroundings pixel and has parameter sigma. Same as mean filter it also blurs the images but it does a better

job in preserving edges of the images.it can be called by using the Gaussian blur function in the Open cv library. Below is the screenshot to show how it is implemented in the code.

```
#apply guassian filter
meanf_x = cv2.GaussianBlur(resized_x, (21,21),0)
meanf_y = cv2.GaussianBlur(resized_y, (21,21),0)
```

Figure 10 Applying Gaussian Filter

5.3. Setup for Experiment 1With out filter On Data set1

The total amount of images used in this experiment is 960 images. Half of them are classic images that are 480 in total, and half of them are thermal images which are 480 in total.

The ratio of training data is 80 percent from the whole data set which are 768 images, and 20% of the data is validation data or test data which is about 192 in total.

The Model has run on the Google Colab which provides an online virtual machine with more computational powers. It has used the Tesla K80 GPU to experiment.

The back bone of the architecture is MOBLIENETV2 and details about the parameters of the model are: the total parameters of the model are 8,047,347. Out of which, the trainable parameters are 8,011,251 and non-train able parameters are 36,096. The number of EPOCS is 200. The train batch size is 32 and Train Steps per EPOCH are 60.

5.3.1. Result and Summary of Experiment 1 (Without Filter)

The model takes about 10,700 seconds to complete the 200 EPOCH. The training loss is 7.98 and the validation loss is 1174.60. The difference between the training loss and validation loss is 1166.62.

This is the sign that the model is under fit and it is because of less trainable parameters and fewer data. When you have a look at the graph below, it shows that the number of EPOCHS should be more then 200 and data should be more. Moreover, when the trainable parameters would be increased and consequently run, the training loss will become less and the difference between training loss and validation loss will also be reduced.

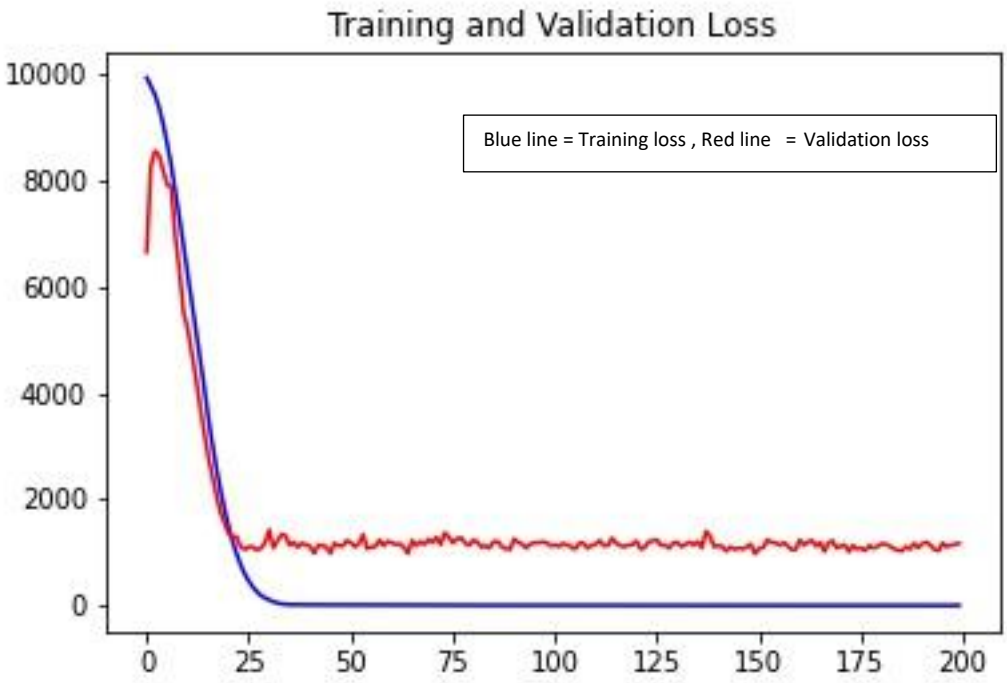

Figure 11 Training and Validation loss on Data set 1 with out Filter

5.3.2. Setup for Experiment 2 with Filter 1 on Data set1

The total amount of images we have used in this experiment is 960 images. Half of them are classic images that are 480 in total, and half of them are thermal images which are 480 in total.

The ratio of training data is 80 percent from the whole data set which are 768 images, and 20% of the data is validation data or test data which is about 192 in total.

In the process of data pre-processing, the mean filter is applied to all train data which is about 768 images in total. The Mean Filter is applied by using the blur function, available in the OPEN CV-library. This function is called by cv2.blur(img,(5*5). This function has two parameters: one is taking an image and the other parameter is the size of the kernel. The researcher has taken the size of 5 for the kernel for this experiment.

The Model runs on the Google Colab which provides an online virtual machine with more computational powers. It uses the Tesla K80 GPU to experiment.

The backbone of the architecture is MOBLIENETV2 and details about the parameters of the model are: the total parameters of the model are 8,047,347. From which the trainable parameters are 8,011,251 and non-train able parameters are 36,096. The number of EPOCS is 200. The train batch size is 32 and training steps per EPOCH are 60.

5.3.3. Result and Summary of Experiment 2 with Filter 1 on Data set 1

The model takes about 11000 seconds to complete the 200 EPOCH. The training loss is 1.8036 and the validation loss is 1179.4733. The difference between the training loss and validation loss is 1,177.6697.

The difference of time between Experiment 1 and Experiment 2 is of 300 seconds and as it can be seen, by applying the mean filter, the value of the training loss is less as compared to the value which we get in the experiment as 7.9890. The difference between these two values is 6.1764, which is very good. This shows that by applying mean filter the training loss could be less and the accuracy of the model can be good, but on the other hand, the difference between training loss and the validation loss is 1,177.4. This shows that the model is under fit and it is because of less trainable parameters and fewer data. It can be seen in the graph below that the number of EPOCHS should be more than 200 and data should be more. Furthermore, if the trainable parameters are increased and consequently run, the training loss will become less and the difference between training loss and validation loss will also be reduced.

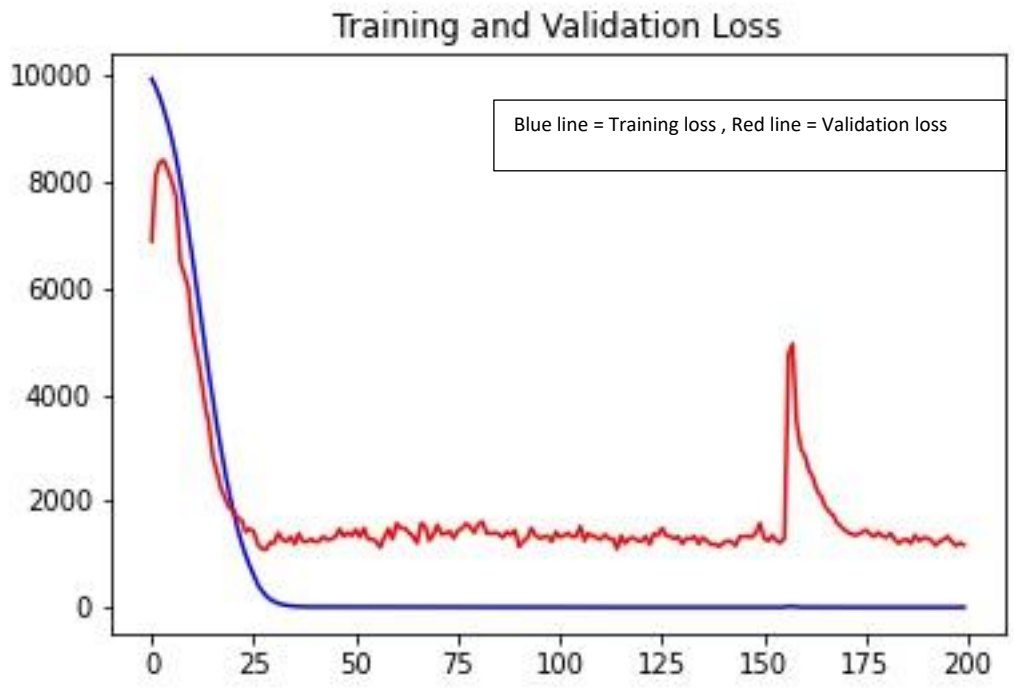

Figure 12 Training and Validation loss on Data set 1 Mean Filter

5.3.4. Setup for Experiment 3 with Filter 2 on Data set 1

The total amount of images used in this experiment is 960 images. Half of them are classic images that are 480 in total, and half of them are thermal images which are 480 in total.

The ratio of training data is 80 percent from the whole data set which are 768 images, and 20% of the data is validation data or test data which is about 192 in total.

In the process of data pre-processing, the median filter is applied to all train data which is about 768 images in total. The Median Filter is applied by using the blur function, available in the OPEN CV-library. This function is called by cv2.medianblur(img, 21). This function has two parameters one is taking the image and the other parameter is the size of the kernel. The size of the kernel must be an odd integer. The researcher has taken the size of 21 for the kernel for this experiment.

The Model runs on the Google Colab which provides an online virtual machine with more computational powers. It uses the Tesla K80 GPU to experiment.

The backbone of the architecture is MOBLIENETV2 and details about the parameters of the model are: the total parameters of the model are 8,047,347 out of which the trainable parameters are 8,011,251 and non-trainable parameters are 36,096. The number of EPOCS is 200. The train batch size is 32 and training steps per EPOCH are 60.

5.3.5. Result and Summary of Experiment 3 Filter 2 on Data set 1

The model takes about 11,600 seconds to complete the 200 EPOCH. The training loss is 2.6523 and the validation loss is 1374.1174. The difference between the training loss and validation loss is 1,371.46.

This experiment takes more time as compared to Experiment 1 and experiment 2. The difference of time between experiment 3 and experiment 1 is 900 seconds and the difference of time between experiment 2 and experiment 3 is 600 seconds which is very high when we apply the median filter.

The training loss of the median filter is less as compared to experiment 1. The difference value between them is 6.09. Interestingly, the difference value of training loss between experiment 2 and experiment 3 is about 0.8523. This shows that the Median Filter helps in reducing the value of the training loss but it takes more time as compared to the mean filter and without filter. However, on the other hand, the difference between training loss and the validation loss is 1,371.46. It shows that the model is under fit and it is because of less trainable parameters and less data. The graph below shows that the number of EPOCHS should be more then 200 and data should be more. Moreover, if the trainable parameters are increased and thus run, the training loss will become less and the difference between training loss and validation loss will also be reduced.

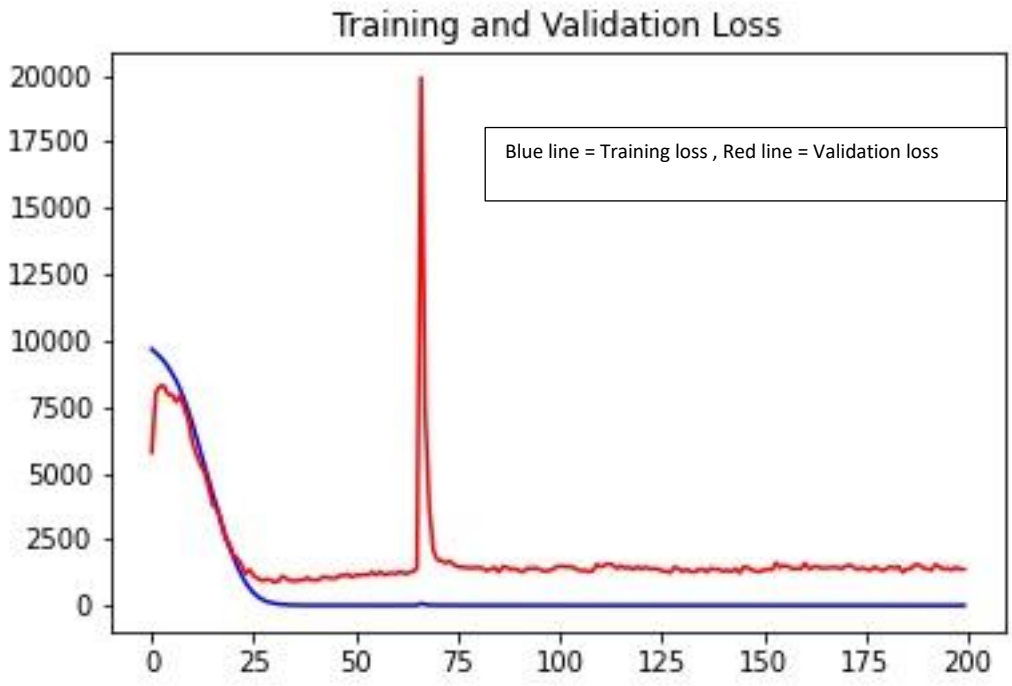

Figure 13 Training and Validation loss on Data set 1 with Median Filter

5.3.6. Setup for Experiment 4 with Filter 3 on Data set 1

The total amount of images used in this experiment is 960 images. Half of them are classic images that are 480 in total, and half of them are thermal images which are 480 in total.

The ratio of training data is 80 percent from the whole data set which are 768 images, and 20% of the data is validation data or test data which is about 192 in total.

In the process of data pre-processing, the Gaussian filter is applied to all train data which is about 1248 images in total. The Mean Filter is applied by using the blur function, available in the OPEN CV-library. This function is called by cv2.Gaussian blur(img, (21,21),0). This function has three parameters: one is taking an image and the other parameter is the size of the kernel. The size of the kernel must be a positive integer. The researcher has taken the size of 21 of the kernel for this experiment. The last parameter is about the sigma value; if it is not given, the filter calculates it from the kernel size.

The Model runs on the Google Colab which provides an online virtual machine with more computational powers. It uses the Tesla K80 GPU to experiment.

The backbone of the architecture is MOBLIENETV2 and details about the parameters of the model are: the total parameters of the model are 8,047,347. Out of which, the trainable parameters are 8,011,251 and non-train able parameters are 36,096. The number of EPOCS is 200. The train batch size is 32 and training steps per EPOCH are 60.

5.3.7. Result and Summary of Experiment 4 with Filter 3 on Data set 1

The model takes about 6,600 seconds to complete the 200 EPOCH. The training loss is 1.4423 and the validation loss is 929.1337. The difference between the training loss and validation loss is 927.6914.

This experiment takes less time as compared to all above three experiments.

The training loss of the Gaussian filter is less as compared to all the above three experiments. The differences among the values of training loss of experiment 1, 2, and 3 are 6.45, 0.36 and 1.21 respectively. This shows that the Gaussian blur filter gives a good result as compared to the mean median and without a filter. However, on the other hand, the difference between training loss and the validation loss is 927.6914. It shows that the model is under fit and it is because of less trainable parameters and fewer data. The graph below shows that the number of EPOCHS should be more than 200, and the data should be more. Moreover, if the trainable parameters are increased and accordingly run, the training loss will be less.

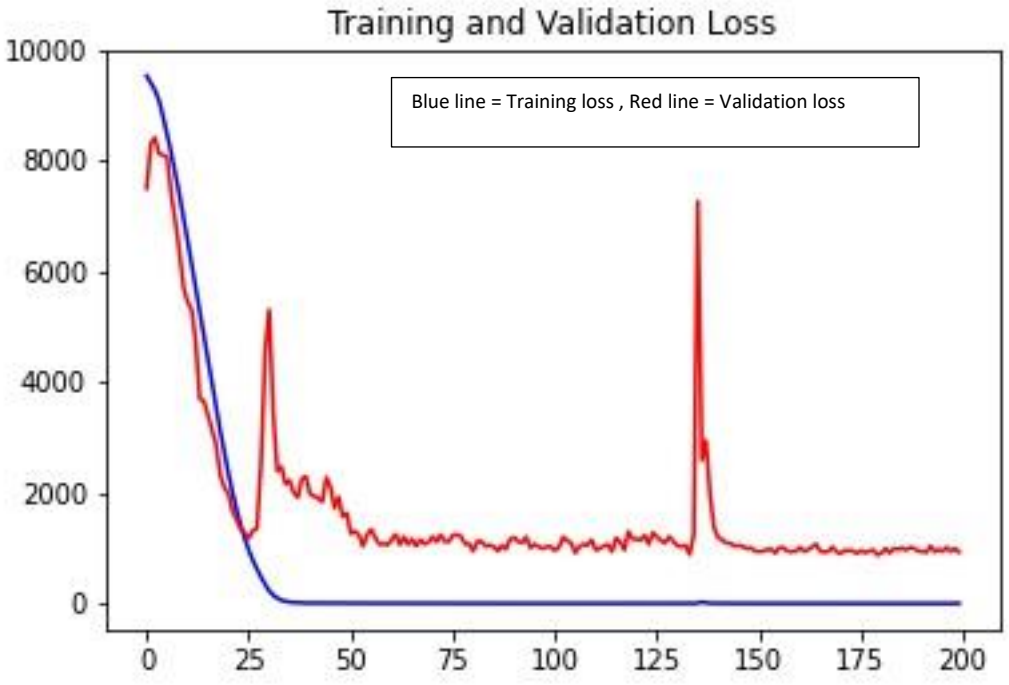

Figure 14 Training and Validation loss on Data set 1 with Gaussian Filter

5.3.8. Summary of 1 to 4 Experiments on Data set 1

The total amount of images we have used in this experiment is 960 images. Half of them are classic images that are 480 in total, and half of them are thermal images which are 480 in total.

The ratio of training data is 80 percent from the whole data set which are 768 images, and 20% of the data is validation data or test data which is about 192 in total.

The back bone of the architecture is MOBLIENETV2 and details about the parameters of the model are: the total parameters of the model are 8,047,347. Out of which, the trainable parameters are 8,011,251 and non-trainable parameters are 36,096.

Type	Training loss	Validation loss	Difference of loss	Total time
Without filter	7.9890	1174.6039	1166.62	10,700seconds
Mean Filter	1.8036	1179.4733	1177.6	11000 seconds
Median filter	2.6523	1374.1174	1371.46	11600 seconds
Gaussian filter	1.4423	929.1337	927.69	6600 seconds

Table 2 Summary of Experiments on Data Set 1

As you can see from the result that with applying filter the value of training loss is reduced

But the difference of the validation loss is very high. Among all filters which the researcher has applied, gaussian filter performs well as compared to all the other filters. The difference between validation loss and training loss is high. This shows that the model is under fit and model requires more data and trainable parameters to have the more good results. Since the researcher is using google Colab, therefore; it cannot be configured as to why there is difference of time as Colab allocates the resources by itself, and they can be changed by it, while the experiment is being executed. Below there are two charts. One chart compares the value of training loss and other chart compares the value of difference of loss.

Figure 15 Training Loss comparison of Data set 1 experiments

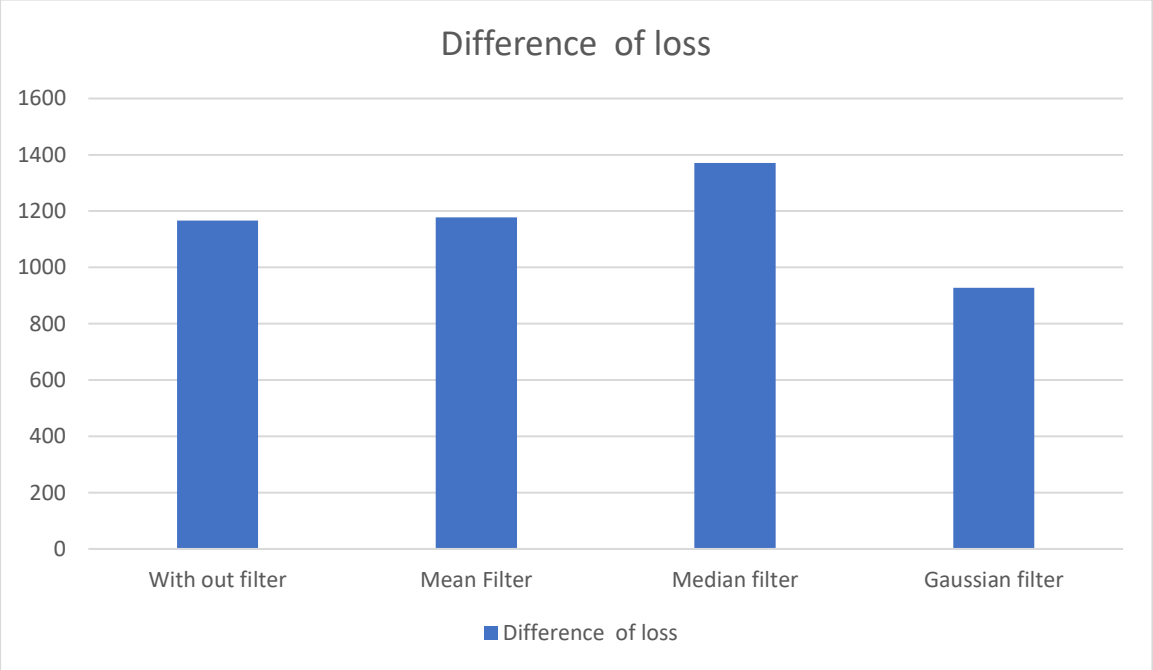

Figure 16 Difference of Validation loss and Training loss on Data Set 1

5.4. Setup for Experiment 1With out filter on Data set 2

The total amount of images used in this experiment is 1200 images. Half of them are classic images that are 600 in total, and half of them are thermal images which are 600 in total.

The ratio of training data is 80 percent from the whole data set which are 960 images, and 20% of the data is validation data or test data which is about 240 in total.

The Model has run on the Google Colab which provides an online virtual machine with more computational powers. It has used the Tesla K80 GPU to experiment.

The backbone of the architecture is MOBLIENETV2 and details about the parameters of the model are: the total parameters of the model are 8,047,347. Out of which, the trainable parameters are 8,011,251 and non-train able parameters are 36,096. The number of EPOCS is 200. The train Batch size is 32 and Train Steps per EPOCH are 60.

5.4.1. Result and Summary of Experiment 1With out Filter on Data set 2

The model takes about 11200 seconds to complete the 200 EPOCH. The training loss is 9.16 and the validation loss is 456.8. The difference between the training loss and validation loss is 447.84.

This is the sign that the model is under fit and it is because of less trainable parameters and fewer data. When you have a look at the graph below, it shows that the number of EPOCHS should be more then 200 and data should be more. Moreover, when the trainable parameters would be increased and consequently run, the training loss will become less and the difference between training loss and validation loss will also be reduced.

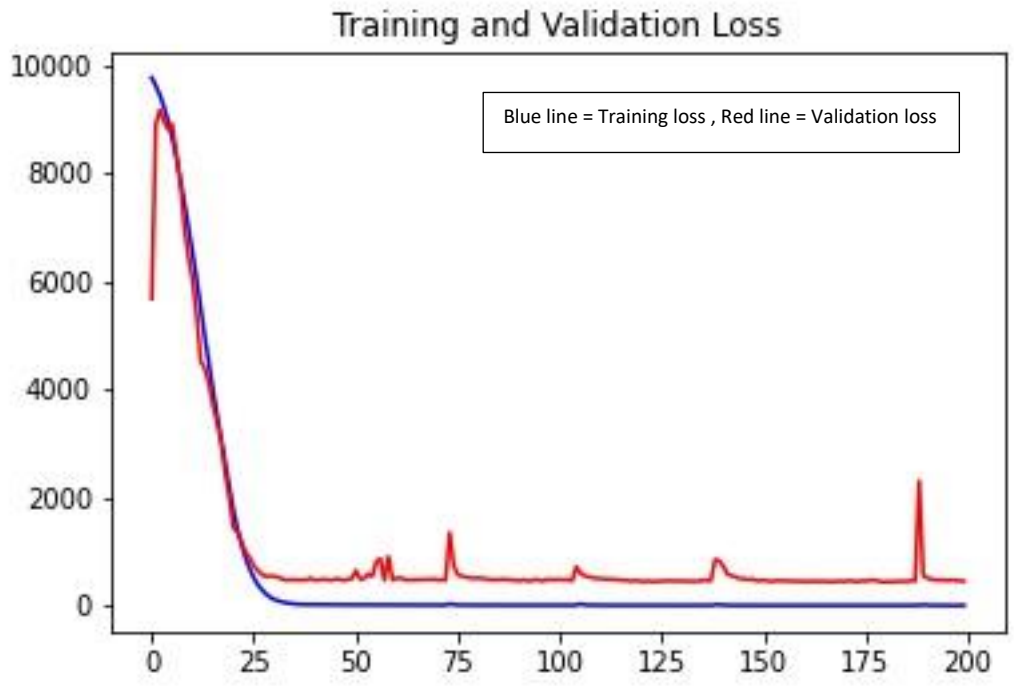

Figure 17 Training and Validation loss on Data set 2 with out Filter

5.4.2. Setup for Experiment 2 with Filter 1 on Data set 2

The total amount of images used in this experiment is 1200 images. Half of them are classic images that are 600 in total, and half of them are thermal images which are 600 in total.

The ratio of training data is 80 percent from the whole data set which are 960 images, and 20% of the data is validation data or test data which is about 240 in total.

In the process of data pre-processing, the mean filter is applied to all train data which is about 960 images in total. The Mean Filter is applied by using the blur function, available in the OPEN CV-library. This function is called by cv2.blur(img,(5*5). This function has two parameters: one is taking an image and the other parameter is the size of the kernel. The researcher has taken the size of 5 for the kernel for this experiment.

The Model runs on the Google Colab which provides an online virtual machine with more computational powers. It uses the Tesla K80 GPU to experiment.

The backbone of the architecture is MOBLIENETV2 and details about the parameters of the model are: the total parameters of the model are 8,047,347. From which the trainable parameters are 8,011,251 and non-train able parameters are 36,096. The number of EPOCS is 200. The train batch size is 32 and training steps per EPOCH are 60.

5.4.3. Result and Summary of Experiment 2 with Filter 1 on Data set 2

The model takes about 11000 seconds to complete the 200 EPOCH. The training loss is 2.706 and the validation loss is 473.83. The difference between the training loss and validation loss is 471.12.

The difference of time between Experiment 1 and Experiment 2 is of 200 seconds and as it can be seen, by applying the mean filter, the value of the training loss is less as compared to the

value which we get in the experiment as 9.16. The difference between these two values is 6.46 which is very good. This shows that by applying mean filter the training loss could be less and the accuracy of the model can be good, but on the other hand, the difference between training loss and the validation loss is 471.12. This shows that the model is under fit and it is because of less trainable parameters and fewer data. It can be seen in the graph below that the number of EPOCHS should be more than 200 and data should be more. Furthermore, if the trainable parameters are increased and consequently run, the training loss will become less and the difference between training loss and validation loss will also be reduced.

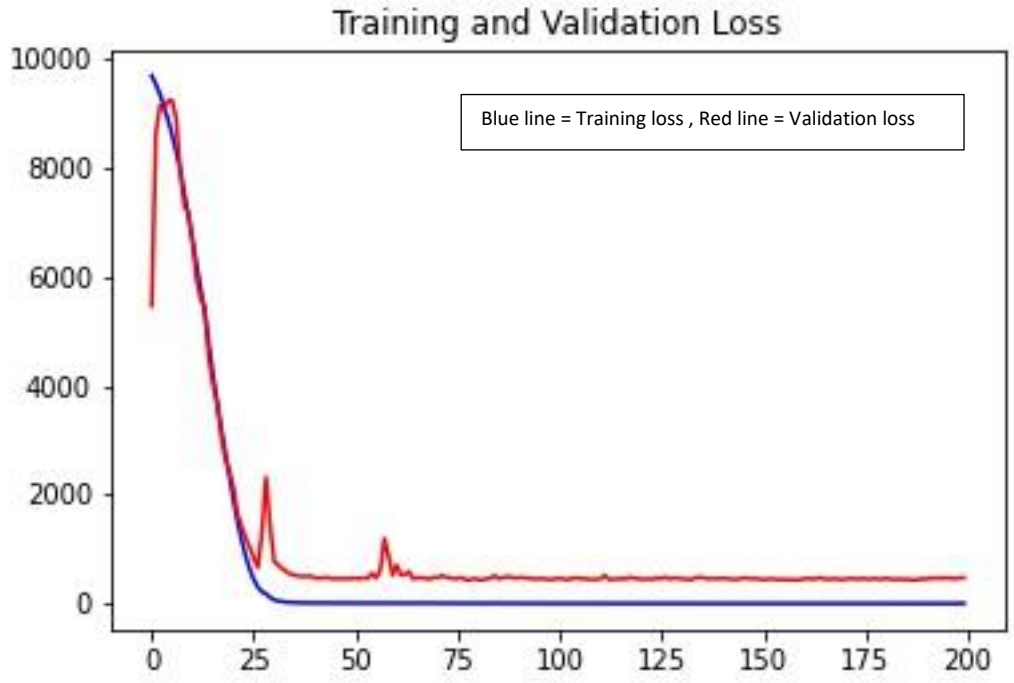

Figure 18 Training and Validation loss on Data set 2 Mean Filter

5.4.4. Setup for Experiment 3 with Filter 2 on Data set 2

The total amount of images used in this experiment is 1200 images. Half of them are classic images that are 600 in total, and half of them are thermal images which are 600 in total.

The ratio of training data is 80 percent from the whole data set which are 960 images, and 20% of the data is validation data or test data which is about 240 in total.

In the process of data pre-processing, the median filter is applied to all train data which is about 960 images in total. The Median Filter is applied by using the blur function, available in the OPEN CV-library. This function is called by cv2.medianblur(img, 21). This function has two parameters one is taking the image and the other parameter is the size of the kernel. The size of the kernel must be an odd integer. The researcher has taken the size of 21 for the kernel for this experiment.

The Model runs on the Google Colab which provides an online virtual machine with more computational powers. It uses the Tesla K80 GPU to experiment.

The backbone of the architecture is MOBLIENETV2 and details about the parameters of the model are: the total parameters of the model are 8,047,347 out of which the trainable parameters are 8,011,251 and non-trainable parameters are 36,096. The number of EPOCS is 200. The train batch size is 32 and training steps per EPOCH are 60.

5.4.5. Result and Summary of Experiment 3 with Filter 2 Data Set 2

The model takes 11600 seconds to complete the 200 EPOCH. The training loss is 2.6523 and the validation loss is 441.65. The difference between the training loss and validation loss is 438.99.

This experiment takes more time as compared to Experiment 1 and experiment 2. The difference of time between experiment 3 and experiment 1 is 400 seconds and the difference of time between experiment 2 and experiment 3 is 600 seconds which is very high when we apply the median filter.

The training loss of the median filter is less as compared to experiment 1. The difference value between them is 6.5. Interestingly, the difference value of training loss between experiment 2 and experiment 3 is about 0.1. This shows that the Median Filter helps in reducing the value of the training loss but it takes more time as compared to the mean filter and without filter. However, on the other hand, the difference between training loss and the validation loss is 438.99. It shows that the model is under fit and it is because of less trainable parameters and less data. The graph below shows that the number of EPOCHS should be more then 200 and data should be more. Moreover, if the trainable parameters are increased and thus run, the training loss will become less and the difference between training loss and validation loss will also be reduced.

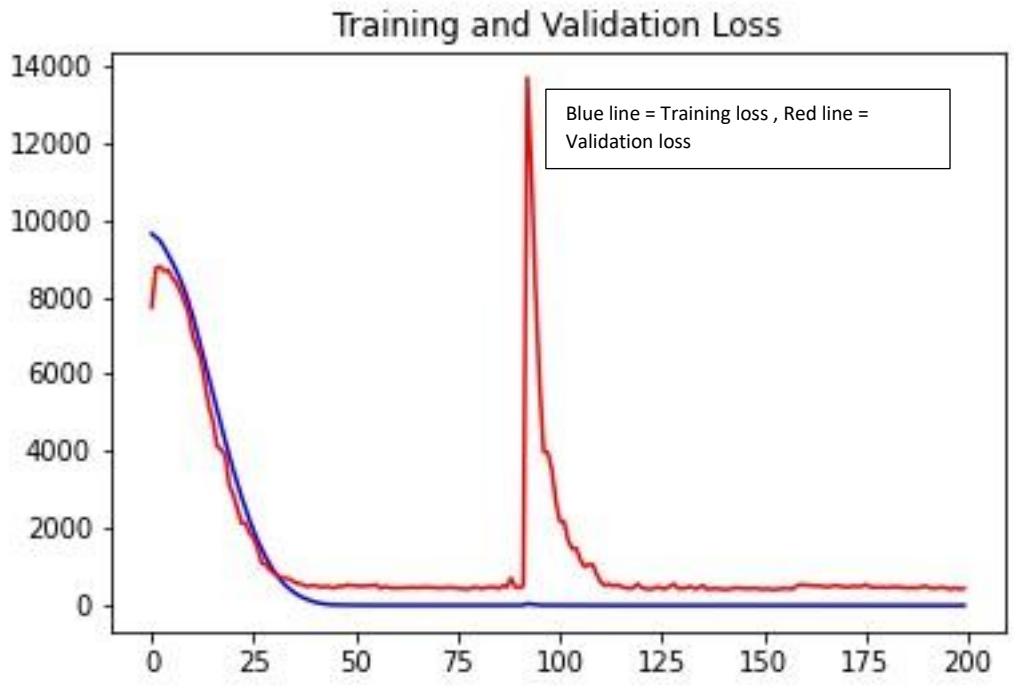

Figure 19 Training and Validation loss on Data set 2 with Median Filter

5.4.6. Setup for Experiment 4 with Filter 3 on Data set 2

The total amount of images used in this experiment is 1200 images. Half of them are classic images that are 600 in total, and half of them are thermal images which are 600 in total.

The ratio of training data is 80 percent from the whole data set which are 960 images, and 20% of the data is validation data or test data which is about 240 in total.

In the process of data pre-processing, the Gaussian filter is applied to all train data which is about 960 images in total. The Mean Filter is applied by using the blur function, available in the OPEN CV-library. This function is called by cv2.Gaussian blur(img, (21,21),0). This function has three parameters: one is taking an image and the other parameter is the size of the kernel. The size of the kernel must be a positive integer. The researcher has taken the size of 21 of the kernel for this experiment. The last parameter is about the sigma value; if it is not given, the filter calculates it from the kernel size.

The Model runs on the Google Colab which provides an online virtual machine with more computational powers. It uses the Tesla K80 GPU to experiment.

The backbone of the architecture is MOBLIENETV2 and details about the parameters of the model are: the total parameters of the model are 8,047,347. Out of which, the trainable parameters are 8,011,251 and non-train able parameters are 36,096. The number of EPOCS is 200. The train batch size is 32 and training steps per EPOCH are 60.

5.4.7. Result and Summary of Experiment 4 with Filter3 on Data Set 2

The model takes about 6,600 seconds to complete the 200 EPOCH. The training loss is 1.4423 and the validation loss is 929.1337. The difference between the training loss and validation loss is 927.6914.

This experiment takes less time as compared to all above three experiments.

The training loss of the Gaussian filter is less as compared to all the above three experiments. The differences among the values of training loss of experiment 1, 2, and 3 are 6.45, 0.36 and 1.21 respectively. This shows that the Gaussian blur filter gives a good result as compared to the mean median and without a filter. However, on the other hand, the difference between training loss and the validation loss is 927.6914. It shows that the model is under fit and it is because of less trainable parameters and fewer data. The graph below shows that the number of EPOCHS should be more than 200, and the data should be more. Moreover, if the trainable parameters are increased and accordingly run, the training loss will be less.

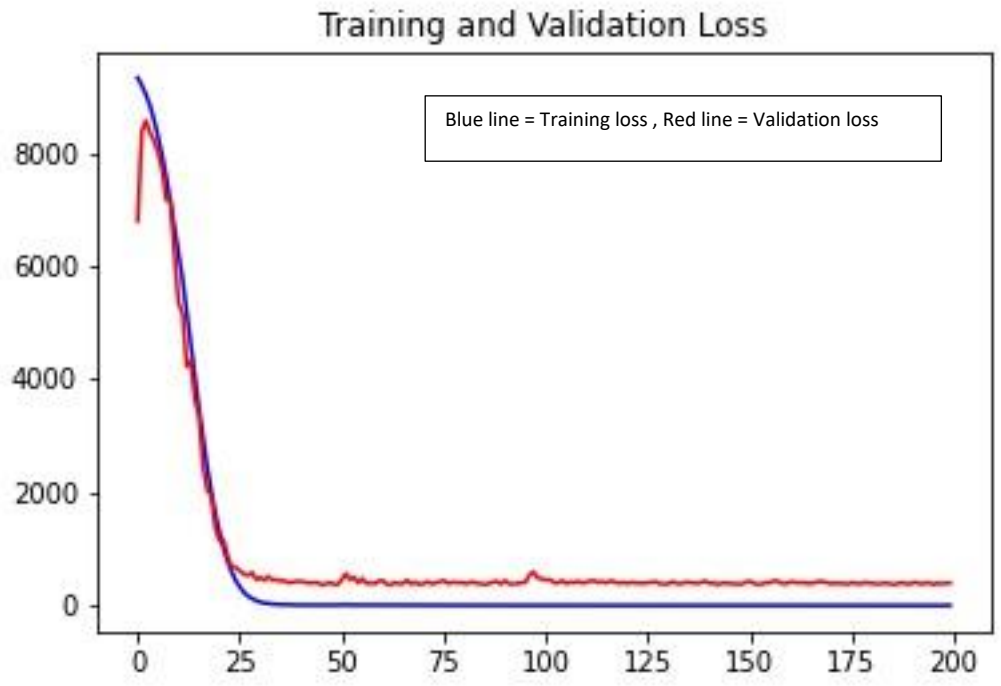

Figure 20 Training and Validation loss on Data set 2 with Gaussian Filter

5.4.8. Summary of 1 to 4 experiments on Data set 2

The total amount of images used in this experiment is 960 images. Half of them are classic images that are 480 in total, and half of them are thermal images which are 480 in total.

The ratio of training data is 80 percent from the whole data set which are 768 images, and 20% of the data is validation data or test data which is about 192 in total.

The back bone of the architecture is MOBLIENETV2 and details about the parameters of the model are: the total parameters of the model are 8,047,347. Out of which, the trainable parameters are 8,011,251 and non-trainable parameters are 36,096.

Type	Training loss	Validation loss	Difference of loss	Total time
Without filter	9.16	456.87	447.71	11,200seconds
Mean Filter	2.7	471.12	468.4	11000 seconds
Median filter	2.6	441.65	439.05	11600 seconds
Gaussian filter	1.3	399.99	398.05	10800 seconds

Table 3 Summary of Experiments on Data set 2

As it can be seen from the result that by applying the filter, the value of training loss is reduced, but the difference of the validation loss is very high. Among all the filters which the researcher has applied, gaussian filter performs good as compared to all the other filters The difference between validation loss and training loss is significantly low as compared to Data set 1 and Data set 3 .This shows that the model is under fit and model requires more data and trainable parameters to have the more good result. Since the researcher is using google Colab therefore, it cannot be configured that why there is a difference of time as Colab allocates the resources by itself and they can be changed by it while the experiment is being executed. Below, there are two bar charts. One chart compares the value of training loss and the other chart compares the value of difference of loss.

Figure 21 Training Loss comparison of Data set 2 experiments

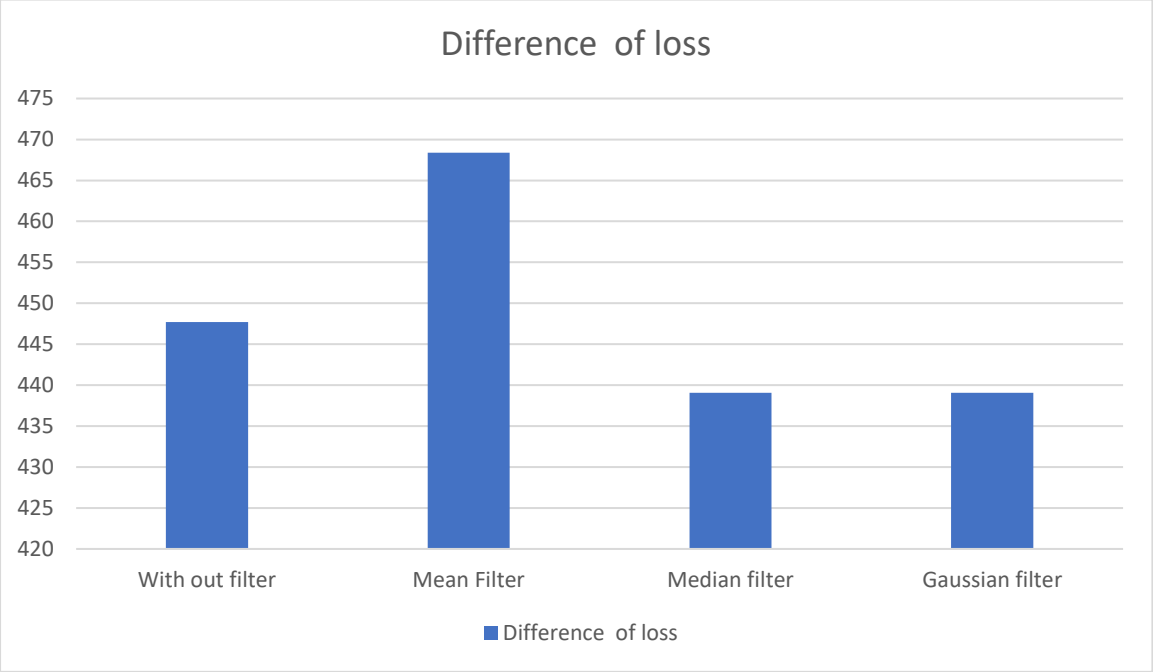

Figure 22 Difference of Validation loss and Training Loss comparison of Data set 2 experiments

5.5. Setup for Experiment 1with out filter on Data set 3

The total amount of images used in this experiment is 1560 images. Half of them are classic images that are 780 in total, and half of them are thermal images which are 780 in total.

The ratio of training data is 80 percent from the whole data set which are 1248 images, and 20% of the data is validation data or test data which is about 312 in total.

The Model has run on the Google Colab which provides an online virtual machine with more computational powers. It has used the Tesla K80 GPU to experiment.

The back bone of the architecture is MOBLIENETV2 and details about the parameters of the model are: the total parameters of the model are 8,047,347. Out of which, the trainable parameters are 8,011,251 and non-train able parameters are 36,096. The number of EPOCS is 200. The train Batch size is 32 and Train Steps per EPOCH are 60.

5.5.1. Result and Summary of Experiment 1 without filter on Data set 3

The model takes about 6331 seconds to complete the 200 EPOCH. The training loss is 10.3086 and the validation loss is 555.87. The difference between the training loss and validation loss is 545.56.

This is the sign that the model is under fit and it is because of less trainable parameters and fewer data. When you have a look at the graph below, it shows that the number of EPOCHS should be more then 200 and data should be more. Moreover, when the trainable parameters would be increased and consequently run, the training loss will become less and the difference between training loss and validation loss will also be reduced.

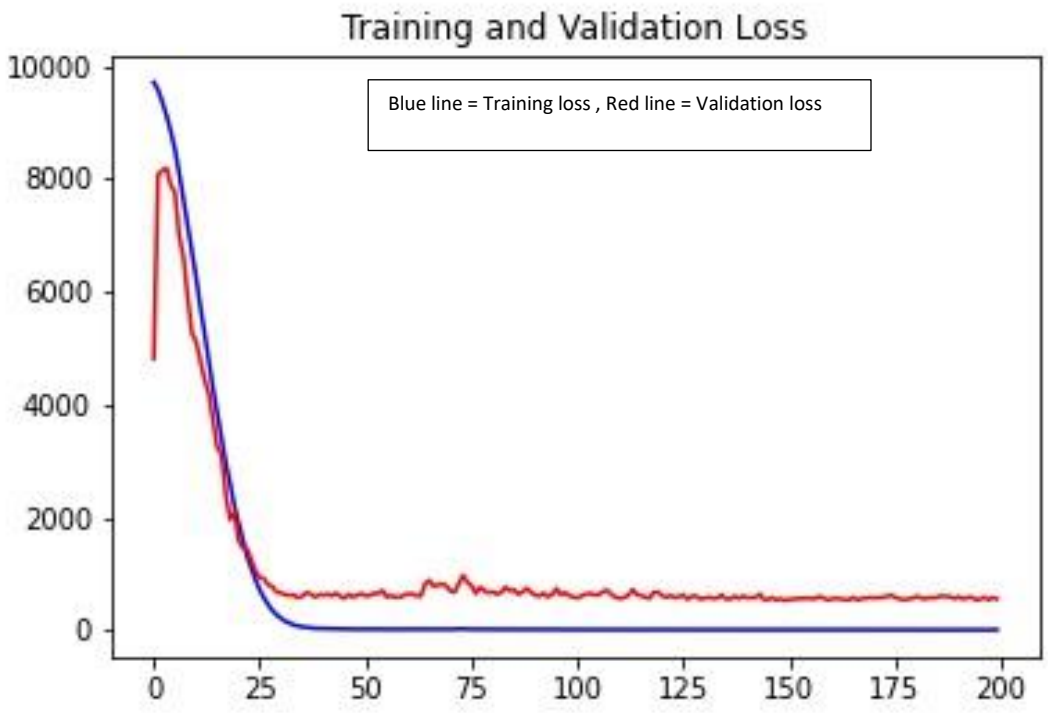

Figure 23 Training and Validation loss on Data set 3 with Out Filter

5.5.2. Setup for Experiment 2 with filter 1 on Data set 3

The total amount of images used for this experiment is 1560 images. Half of them are classic images that are 780 in total, and half of them are thermal images which are 780 in total.

The ratio of training data is 80 percent from the whole data set which are 1248 images and 20% of the data is validation data or test data which is about 312 in total.

In the process of data pre-processing, the mean filter is applied to all train data which is about 1248 images in total. The Mean Filter is applied by using the blur function, available in the OPEN CV-library. This function is called by cv2.blur(img,(5*5). This function has two parameters: one is taking an image and the other parameter is the size of the kernel. The researcher has taken the size of 5 for the kernel for this experiment.

The Model runs on the Google Colab which provides an online virtual machine with more computational powers. It uses the Tesla K80 GPU to experiment.

The backbone of the architecture is MOBLIENETV2 and details about the parameters of the model are: the total parameters of the model are 8,047,347. From which the trainable parameters are 8,011,251 and non-train able parameters are 36,096. The number of EPOCS is 200. The train batch size is 32 and training steps per EPOCH are 60.

5.5.3. Result and Summary of Experiment 2 Filter 1 on Data set 3

The model takes about 6000 seconds to complete the 200 EPOCH. The training loss is 3.2645

and the validation loss is 606.614. The difference between the training loss and validation loss is 603.60.

The difference of time between Experiment 2 and Experiment 1 is of 331 seconds and as it can be seen, by applying the mean filter, the value of the training loss is less as compared to the value which we get in the experiment as 10.30. The difference between these two values is 7.30, which is very good. This shows that by applying mean filter the training loss could be less and the accuracy of the model can be good, but on the other hand, the difference between training loss and the validation loss is 603.60. This shows that the model is under fit and it is because of less trainable parameters and fewer data. It can be seen in the graph below that the number of EPOCHS should be more than 200 and data should be more. Furthermore, if the trainable parameters are increased and consequently run, the training loss will become less and the difference between training loss and validation loss will also be reduced.

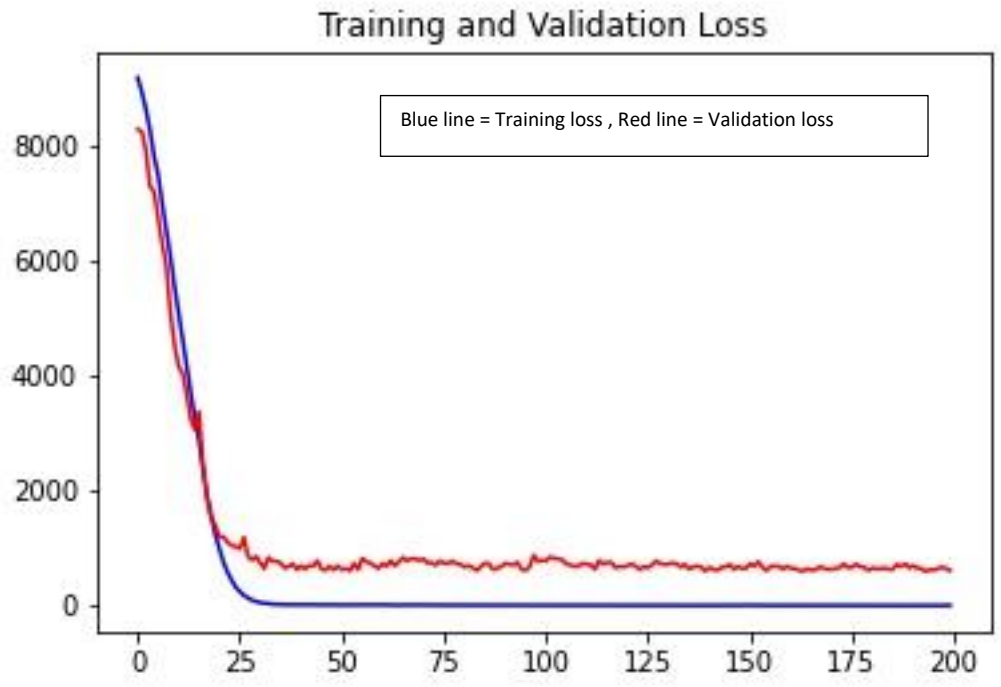

Figure 24 Training and Validation loss on Data set 3 with Mean Filter

5.5.4. Setup for Experiment 3 with Filter 2 with data set 3

The total amount of images used for this experiment is 1560. Half of them are classic images: 780 in total and half of them are thermal images which are 780 in total. The ratio of train data is 80 percent from the whole data set which are 1248 images and 20 percent of the data is validation data or test data which is about 312 in total.

In the process of data pre-processing, the median filter is applied to all train data which is about 1248 images in total. The Mean Filter is applied by using the blur function, available in the OPEN CV-library. This function is called by cv2.medianblur(img, 21). This function has two

parameters one is taking the image and the other parameter is the size of the kernel. The size of the kernel must be an odd integer. The researcher has taken the size of 21 for the kernel for this experiment.

The Model runs on the Google Colab which provides an online virtual machine with more computational powers. It uses the Tesla K80 GPU to experiment.

The backbone of the architecture is MOBLIENETV2 and details about the parameters of the model are: the total parameters of the model are 8,047,347 out of which the trainable parameters are 8,011,251 and non-trainable parameters are 36,096. The number of EPOCS is 200. The train batch size is 32 and training steps per EPOCH are 60.

5.5.5. Result and Summary of Experiment 3 with Filter 2 on Data set 3

The model takes about 12243 seconds to complete the 200 EPOCH. The training loss is 4.2998 and the validation loss is 856.74. The difference between the training loss and validation loss is 852.41.

This experiment takes more time as compared to Experiment 1 and experiment 2. The difference of time between experiment 3 and experiment 1 is 5921 seconds and the difference of time between experiment 6 and experiment 3 is 5581 seconds which is very high when we apply the median filter.

The training loss of the median filter is less as compared to experiment 1. The difference value between them is 6.09. Interestingly, the difference value of training loss between experiment 6 and experiment 7 is about 1.02. This shows that the Median Filter helps in reducing the value of the training loss but it takes more time as compared to the mean filter and without filter. However, on the other hand, the difference between training loss and the validation loss is 852.60. It shows that the model is under fit and it is because of less trainable parameters and less data. The graph below shows that the number of EPOCHS should be more then 200 and data should be more. Moreover, if the trainable parameters are increased and thus run, the training loss will become less and the difference between training loss and validation loss will also be reduced.

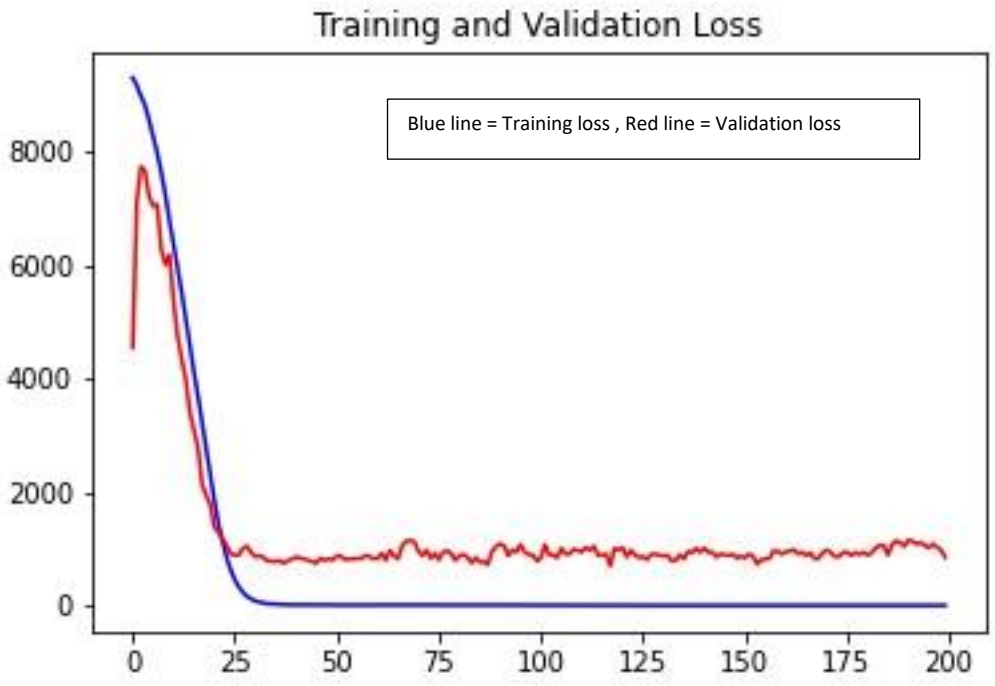

Figure 25 Training and Validation loss on Data set 3 with Median Filter

5.5.6. Setup for Experiment 4 with Filter 3 on data set 3

The total amount of images used for this experiment is 1560 images. Half of them are classic images that are 780 in total and half of them are thermal images and they are 780 in total.

The ratio of training data is 80 percent from the whole data set which are 1248 images and 20 percent of the data is validation data or test data which is about 312 in total.

In the process of data pre-processing, the Gaussian filter is applied to all train data which is about 1248 images in total. The Mean Filter is applied by using the blur function, available in the OPEN CV-library. This function is called by cv2.Gaussian blur(img, (21,21),0). This function has three parameters: one is taking an image and the other parameter is the size of the kernel. The size of the kernel must be a positive integer. The researcher has taken the size of 21 of the kernel for this experiment. The last parameter is about the sigma value; if it is not given, the filter calculates it from the kernel size.

The Model runs on the Google Colab which provides an online virtual machine with more computational powers. It uses the Tesla K80 GPU to experiment.

The backbone of the architecture is MOBLIENETV2 and details about the parameters of the model are: the total parameters of the model are 8,047,347. Out of which, the trainable parameters are 8,011,251 and non-train able parameters are 36,096. The number of EPOCS is 200. The train batch size is 32 and training steps per EPOCH are 60.

5.5.7. Result and Summary of Experiment 4 with Filter 3 on Data set 3

The model takes about 10600 seconds to complete the 200 EPOCH. The training loss is 1.5049 and the validation loss is 741.320. The difference between the training loss and validation loss is 739.80.

This experiment takes more time as compared to Experiment 1 experiment 2 but takes less time from experiment 3. The difference of time between experiment 4 and experiment 8 is 4296 seconds and difference of time between experiment 3 and experiment8 is 1643 seconds.

The training loss of the Gaussian filter is less as compared to all the above three experiments. The differences among the values of training loss of experiment 1, 2, and 3 are 8.8, 1.7 and 2.79 respectively. This shows that the Gaussian blur filter gives a good result as compared to the mean median and without a filter. It takes more time than experiment 1 and experiment 2 but the training loss value is lowest as compared to all the other experiments. However, on the other hand, the difference between training loss and the validation loss is 739.80. It shows that the model is under fit and it is because of less trainable parameters and fewer data. The graph below shows that the number of EPOCHS should be more than 200, and the data should be more. Moreover, if the trainable parameters are increased and accordingly run, the training loss will be less.

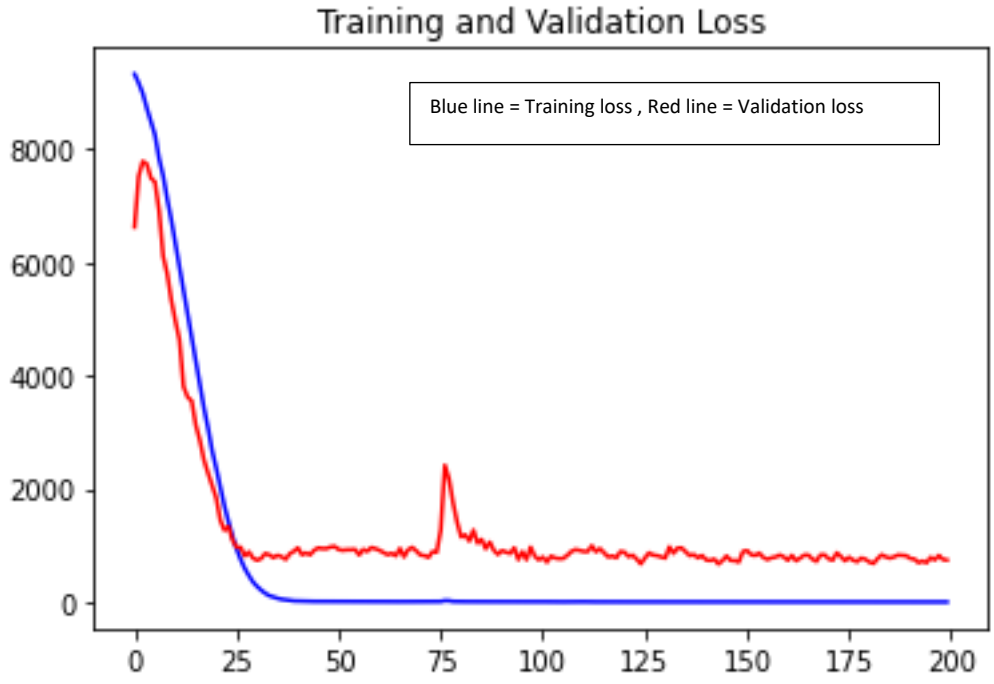

Figure 26 Training and Validation loss on Data set 3 with Gaussian Filter

5.5.8. Summary of 1 to 4 experiments on Data set 3

In these experiments, the total amount of images used is 1560 images. Half of them are classic images that are 780 in total and half of them are thermal images which are 780 in total.

The ratio of training data is 80 percent from the whole data set which are 1248 images and 20 percent of the data is validation data or test data which is about 312 in total.

The back bone of the architecture is MOBLIENETV2 and details about the parameters of the model are: the total parameters of the model are 8,047,347. Out of which, the trainable parameters are 8,011,251 and non-trainable parameters are 36,096.

Type	Training loss	Validation loss	Difference of loss	Total time
Without filter	10.308	555.87	545.4	6331 seconds
Mean Filter	3.26	606.61	603.41	6000 seconds
Median filter	4.29	856.74	853.48	12243 seconds
Gaussian filter	1.50	741.32	739.82	10600 seconds

Table 4 Summary of experiments on Data set 3

As you can see from the result that with applying filter the value of training loss is reduced

But the difference of the validation loss is very high. Among all filters which researcher has applied gaussian filter performs good as compare to all other filters the graphical representation you can see in the bar chart. The value of training loss is less if we compared it with other values which are with filters.

Figure 27 Training Loss comparison of Data set 3 experiments

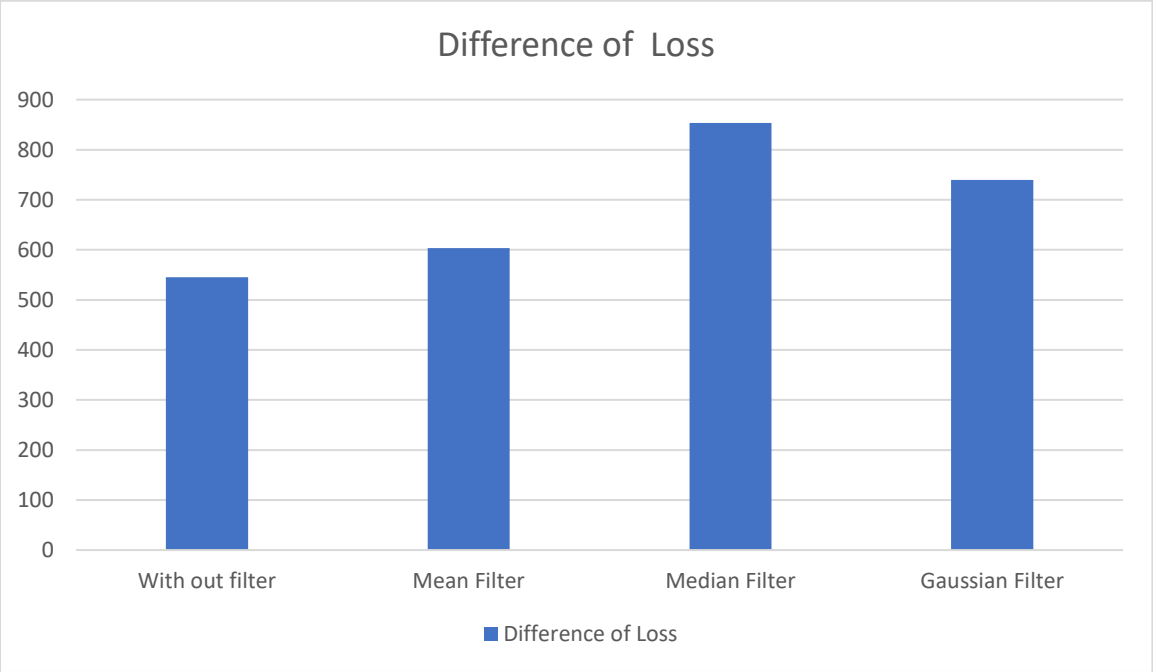

Figure 28 Difference of Validation loss and Training Loss comparison of Data set 3 experiments

6. Conclusion and Future work

In This chapter researcher going to discuss about the results of all experiments and about the future work that what can be more done

6.1. Conclusion of All experiments

By applying filters to the data in the step of pre-processing of data, the value of loss is reduced on each data set .This indicates that by applying filters to the data in step of pre-processing the accuracy can be enhanced because we are evaluating the model on the value of training loss. When model is fully accurate or trained the loss value of training loss should be 0.01 or plus minus between them and value of validation loss should be equal to the value of training loss. But in the researcher's case, the difference between training loss and validation loss is higher. When the difference between validation loss and training loss is high the model is underfit .This due to many reasons some of them are less amount of data and less amount of train able parameters. The results in our experiment shows that when we increase the amount of data set the difference between training loss and validation is reduced which is the positive impact on the performance of the model. Below there are two chart. One bar chart shows the comparison value of training loss of three data sets with filters and second shows the difference of validation loss among three data sets.

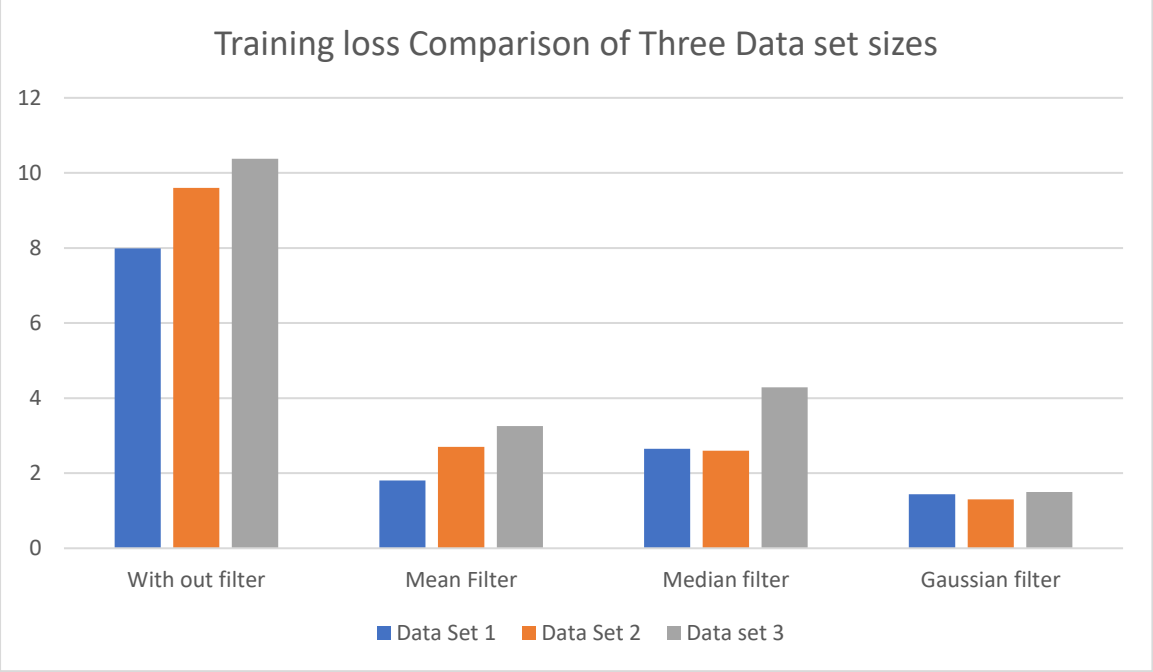

Figure 29 Training loss comparison of all experiment

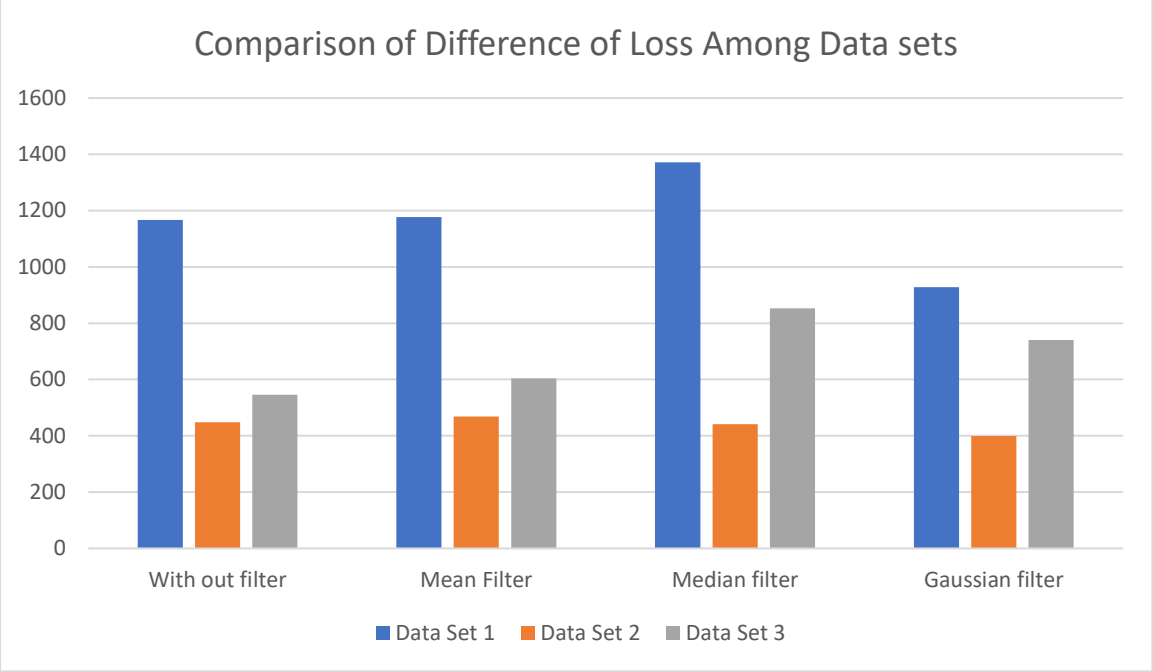

Figure 30 Comparison of Difference loss Among Data Sets

6.2. Future work

For the future work the researcher can do the same above experiments but with more trainable parameters which were mentioned by the Ekenel, 2019 and also use the same amount of data which was 4960 images in total. Hope fully when the researcher runs that on full parameters the accuracy will be enhanced and thermal images can be recognized more efficiently.

We can also use some combination of filters on the data set and can see that what impact it has on the model accuracy and among them which combination will gave better results.

One can perform the same set experiments on other data sets which are available so the results. For example, other data set which contains images in different poses of the face and can check what results we get them after apply filters in the step of pre-processing.

6.3. Self-Reflection

By doing this project I learned a lot of new things during this process. First of all the domain is new itself which is deep learning. Before starting the project, I had no idea about the deep learning and its sub domain which are convolutional neural network and things which are related to this. I learned these new techniques and it self it was an achievement for me to implement some one work by myself and addressing the issue of the gap ,find a new thing which was about the filters impact on the data . I faced a lot of technical issues in the environment till the deploying of the model. But I learned and I firmly believe that my skills are now enhanced in the domain of deep learning and dealing with the images . I ready to step out in the world to secure a role for me as Deep learning engineer in the images .

References

Ekenel, A. K. (2019). Thermal to visible Face recognition Using Deep autoencoders. *International Conference of the Biometrics Special Interest Group (BIOSIG), Darmstadt, Germany, 2019,*, 1-5.

K. Mallat, N. D. (2019). Cross-spectrum thermal to visible face recognition based on cascaded image. *2019 International Conference on Biometrics (ICB)*, 8.

Lin, S. &. (2019). Thermal Face Recognition under disguised conditions. *inproceedings{*, 1-7.

Samah A.F.Mansoor, S. S. (2019). TRI FACE NET: Thermal IR Facial Recognition. *2102 ,12th International Congress on Image and Signal Processing , BioMedical Engineering and Informatics (CISP_BMEI)*.

X. Dong, K. W. (2019). A Secure Visual-thermal Fused Face Recognition System Based on Non-linear Hashing. *2019 IEEE 21st International Workshop on Multimedia Signal Processing (MMSP), Kuala Lumpur, Malaysia*, 1-6.

Yakubovskiy, P. (2019). Segmentation Models. *GitHub repository*.

Appendences

Installing Segmentation model

In this part, the segmentation model library is installed and their dependency libraries, which are made by Yakubovskiy, 2019.

```
!pip install segmentation-models

pip uninstall keras

!pip install Keras==2.4.0
```

Mount the Google Drive

```
from google.colab import drive

drive.mount('/content/gdrive')
```

Importing Other libraries

```
 import segmentation_models as sm

 import tensorflow as tf

  from glob import glob

  import matplotlib.pyplot as plt

  import cv2

  import os

  import numpy as np

   import matplotlib.image as mp
```

Generating Samples

```
classic = glob("/content/gdrive/My Drive/ThermalImageTest/Database1/*/*/classic")

thermal = glob("/content/gdrive/My Drive/ThermalImageTest/Database1/*/*/thermal_bmp")

X = list()

Y = list()


for subj in range(len(classic)):
```

```
    X.extend(sorted(glob(classic[subj] + '/*.bmp')))
    Y.extend(sorted(glob(thermal[subj] + '/*.bmp')))

sample = np.random.randint(0, len(X))
X[sample], Y[sample]
```

Splitting The data

```
split = 0.8
train_X = X[:int(len(X)*split)]
train_Y = Y[:int(len(Y)*split)]

val_X = X[int(len(X)*split):]
val_Y = Y[int(len(X)*split):]

print("Lengths Train Data:")
print(len(train_X), len(train_Y))
print("Lengths Val Data:")
print(len(val_X), len(val_Y))
```

Data Set Generator Class

```
class Dataset():
    def __init__(self, X, Y, BATCH_SIZE):
        autotune = tf.data.experimental.AUTOTUNE
        shuffle_buffer_size = len(X)

        input_tensors = tf.data.Dataset.from_tensor_slices((X,Y))
        input_tensors = input_tensors.shuffle(shuffle_buffer_size).repeat(-1)
        input_tensors = input_tensors.map(map_func=self.parse_function, num_parallel_calls=a
utotune)
        self.input_tensors = input_tensors.batch(batch_size=BATCH_SIZE).prefetch(buffer_siz
e=autotune)
```

```
        self.feed = iter(self.input_tensors)

    def parse_function(self, x, y):
        def _parse_function(_x, _y):
            _x = _x.decode('UTF-8')
            _y = _y.decode('UTF-8')

            image = cv2.imread(_x)
            label = cv2.imread(_y)

            resized_x = cv2.resize(image, (224,224))
            resized_y = cv2.resize(label, (224,224))
            #Apllying Mean filter
            #meanf_x = cv2.blur(resized_x,(5,5))
            #meanf_y = cv2.blur(resized_y,(5,5))
            #apply median filter
            #meanf_x = cv2.medianBlur(resized_x, 21)
            #meanf_y = cv2.medianBlur(resized_y, 21)
            #apply guassian filter
            meanf_x = cv2.GaussianBlur(resized_x, (21,21),0)
            meanf_y = cv2.GaussianBlur(resized_y, (21,21),0)

            return meanf_x.astype(np.float32), meanf_y.astype(np.float32)

        image, label = tf.numpy_function(_parse_function, [x, y], [tf.float32, tf.float32])

        return image, label
```

Changing Segmentation Model to Encoder Decoder

```
BACKBONE = 'resnet18'
```

```
model = sm.Unet(BACKBONE, input_shape = (224, 224, 3))


model.layers.pop()
model.layers.pop()

x = tf.keras.layers.Conv2D(3, (1,1))(model.layers[-1].output)
model = tf.keras.models.Model(model.layers[0].output, x)

model.compile(optimizer='adam', loss='mean_squared_error')

model.summary()
```

Setting Configuration of Model

```
EPOCHS = 200
TRAIN_BATCH_SIZE = 32
TRAIN_STEPS_PER_EPOCH = 60
VAL_BATCH_SIZE = 32
VAL_STEPS_PER_EPOCH = 15

dataset_train = Dataset(train_X, train_Y, TRAIN_BATCH_SIZE)
dataset_val = Dataset(val_X, val_Y, VAL_BATCH_SIZE)

train_generator = dataset_train.input_tensors
val_generator = dataset_val.input_tensors
```

Model Training

```
weights_filepath = "/content/gdrive/My Drive/ThermalImageTest/weights/model_{epoch:02d
}-{val_loss:.2f}.hdf5"

model_callbacks = tf.keras.callbacks.ModelCheckpoint(
    filepath = weights_filepath,
    save_weights_only=False,
    save_best_only=False
```

```
)

history = model.fit_generator(
    train_generator, steps_per_epoch=TRAIN_STEPS_PER_EPOCH, epochs=EPOCHS,
    validation_data=val_generator, validation_steps=VAL_STEPS_PER_EPOCH,
    verbose=1, callbacks=[model_callbacks], shuffle=True
)
```

Plotting the Graph

```
def plot_training(history):
    train_loss = history.history['loss']
    val_loss = history.history['val_loss']
    epochs = range(len(train_loss))

    plt.plot(epochs, train_loss, 'b')
    plt.plot(epochs, val_loss, 'r')
    plt.title('Training and Validation Loss')
    plt.savefig('/content/gdrive/My Drive/ThermalImageTest/loss_results1200Gaussian.jpg')
    plt.show()


plot_training(history)
```