SEA ICE CLASSIFICATION

A PROJECT REPORT

Submitted by

DIP PATEL (19CP008)

In partial fulfillment for the award of the degree of

B. TECH. in COMPUTER ENGINEERING

4CP33: Full Semester External Project (FSEP)

BIRLA VISHVAKARMA MAHAVIDYALAYA
(ENGINEERING COLLEGE)
(An Autonomous Institution)
VALLABH VIDYANAGAR

Affiliated to

GUJARAT TECHNOLOGICAL UNIVERSITY, AHMEDABAD

Academic Year: 2022 – 2023

BVM ENGINEERING COLLEGE, VALLABH VIDYANAGAR-388120

APPROVAL SHEET

The project work entitled "**SEA ICE CLASSIFICATION**" carried out by **Dip Patel (19CP008)** is approved for the submission in the course **4CP33, Full Semester External Project** for the partial fulfillment for the award of the degree of B. Tech. in Computer Engineering.

Date:

Place:

Signatures of Examiners:

(Names and Affiliations)

CERTIFICATE

This is to certify that Project Work embodied in this project report titled "**SEA ICE CLASSIFICATION**" was carried out by **DIP PATEL (19CP008)** under the course **4CP33, Full Semester External Project** for the partial fulfillment for the award of the degree of B. Tech. in Computer Engineering. Followings are the supervisors at the student:

Date:

Place:

J. B. Modi	N. M. Patel	M. I. Hasan
SCI/ENGR-SG	Professor	Assistant Professor
SIPG-MDPD/ISRO	Computer Engineering Department,	Computer Engineering Department,
Ahemdabad	BVM Engineering College	BVM Engineering College

Dr. Darshak G Thakore
Prof. & Head,
Computer Engineering Department,
BVM Engineering College

COMPUTER ENGINEERING DEPARTMENT, BVM ENGINEERING COLLEGE, VALLABH VIDYANAGAR-388120

DECLARATION OF ORIGINALITY

I hereby certify that I am the sole author of this report under the course 4CP33 (Full Semester External Project) and that neither any part thereof nor the whole of the report has been submitted for a degree to any other University or Institution.

I certify that, to the best of my knowledge, the current report does not infringe upon anyone's copyright nor does it violate any proprietary rights and that any ideas, techniques, quotations or any other material from the work of other people included in my report, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the boundary of fair dealing within the meaning of the Indian Copyright (Amendment) Act 2012, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in the current report and have included copies of such copyright clearances to the appendix of this report.

I declare that this is a true copy of report, including any final revisions, as approved by the report review committee.

I have checked write-up of the present report using anti-plagiarism database and it is in permissible limit. However, at any time in future, in case of any complaint pertaining of plagiarism, I am the sole responsible person for the same. I also understand that, in case of such complaints of plagiarism, as per the University Grants Commission (UGC) norms, the University can even revoke the degree conferred to the student submitting such a plagiarized report.

Date:

Institute Code: 007

Institute Name: Birla Vishvakarma Mahavidyalaya (BVM) Engineering College

Dip Patel

19CP008

ACKNOWLEDGEMENT

First I would like to thank Ramanujam sir(Head of MDPD dept.) for giving me this wonderful opportunity to work them. Also, I would like to express my sincere gratitude to my industry guide Jalpa B. Modi and Wasim Akaram for their guidance and reviewing my report. Also thanks to Yatharth M. Sant for his technical guidance and Mayur D. Chopda for encouraging me.

I would like to express our sincere gratitude to my project guides Dr. N. M. Patel and Prof. M. I. Hasan for their guidance and support throughout the project. Project ideas tips from both the guide that what I can do further to the project had provided us great motivation. Their expertise and invaluable advice have been instrumental in the successful completion of this project.

I am motivated to improve my presentation skills by their advices. Their insights and suggestions were greatly appreciated and helped to improve the quality of the project and this report. I am also motivated to write a research paper on the project which I will definitely do in the future by our mentors.

Dip Patel

19CP008

Plagiarism Report

Introduction

Literature study

Implementation

Results

Appendix

Table of Contents

List of Figures

List of Tables

Abstract

Due to increase in global warming, melting of sea ice at polar region becomes faster than ever. In that case, if we have some way through which we could measure situation of ice at polar region, then that can be useful. Therefore, in this project, we carried out sea ice classification of SAR (Synthetic Aperture Radar) images which are captured over Hudson bay(In Canada). In this study, spatiotemporal dataset of Sentinel-1 SAR, named SI-STSAR-7 is used. Here, we have used deep learning technique to solve this problem. Resnet and convLSTM are the two model that we have used in our study. Resnet is used over other CNN architecture because Resnet works well on images having same features size (which is not the case in inception model family). convLSTM is second model that we have used because dataset contains temporal feature. So, to leverage that feature we have used convLSTM. In both architectures, we got more than 94% testing accuracy. After model training our task is to automate entire process from downloading product form internet to carry out all pre-processing steps and generate prediction. For downloading we wrote the shell script which download SAR images from ASF(Alaska satellite facility) vertex. Preprocessing steps were carried out in SNAP toolkit. Lastly, we generate output in two form, First is image which depict which type of ice available in particular area while second is bar chart which represent percentage proportion of different ice-type.

Chapter 1

Introduction

1.1 Overview

Monitoring the sea ice at polar region is necessary due to the increase in global warming. Scientist have noted that ice at polar region is melting at faster pace then ever, so in this situation it will be helpful if somehow we could measure what is the condition of ice at polar region. Sea ice classification will provide us a way through which we can measure condition of ice on polar region, hence in this study we perform sea ice classification on Hudson bay.

Before this, many studies have been carried out, in which they have used different image processing techniques like Gabor wavelet technique [1], while other used machine learning and deep learning techniques like Neural Network [2, 3], Support vector machine(SVM), Markov random field(MRF).

In this study, we used deep learning technique to classify SAR(synthetic aperture radar) image into different ice category. Primarily two model we used named ConvLSTM and Resnet. ConvLSTM is used because in dataset we have temporal resolution available as well and Resnet is used because it is state of the art model that work quite well on image processing task.

The SI-STSAR-7 training dataset available on IEEE data portal has been used for development of our models.

Active microwave sensor like SAR have ability to observe the earth all day without the limitation of light. Also, microwave have ability to penetrate through cloud. SAR sensor emits the microwaves. These waves will interact with earth surface and sub-surface. That incident beam may get absorbed or may get backscatter in all directions at surface level. Some waves may get through the surface and later backscatter through inner layer of surface. Microwaves that are backscattered in the direction of radar antenna are only captured by radar. Backscattered waves which are not in the direction of antenna are lost and radar does not have any information about that. Radar have information only for those waves which backscattered in the direction of antenna.

Different types of ice have different back-scattering coefficients. This information would be helpful to classify images. There are some studies in which some threshold values have been calculated for each ice types using statistics. But there is significant overlap in the calculated threshold values so those method are not much reliable. Therefore, it is better to use deep learning method, which in-turn finds best features from images that are best representative of particular sea ice types, and then model uses it to make prediction.

1.2 Motivation

Nowadays, due to global warming man kind faces many difficulties like increase in global temperature, climate change, flooding in some part of world etc. Due to increase in global temperature, melting of sea ice at polar region now become faster than ever. Ice melting can severely effect activity of human kind like ship navigation, ice bed mining, ocean hydrological condition etc. For this reason if we have some way through which we can measure the sea condition of polar region then it would become helpful to us. For this reason I decided to work on this project in which we can detect type of ice from SAR(Synthetic Aperture Radar) image.

SAR image is different form the optical image. SAR image looks like grey-scale image but it is more than that. SAR satellite basically use microwaves and microwaves can easily penetrate through clouds as well as rain fall. And It is also a active sensor hence, there is no requirement of sun light at all. In addition to this, Microwaves can penetrate though sub-surface and latter reflect from inner layer of soil or ice so this characteristic will become useful to determine which type of ice. Due to this two benefit over optical we preferred to use SAR images over optical image.

Chapter 2

Literature survey

Many studies on sea ice classification have been carried out using Gabor wavelet techniques [4], Markov random field (MRF) [5, 6], neural network [7, 8], support vector machine (SVM) [9, 10], and other methods. With the widespread application of deep learning in the image field, researchers have applied deep learning models to sea ice classification and achieved high classification accuracy [11, 12]. However, the performance of deep learning methods depends on a large dataset.At present, most sea ice analysis methods are typically based on a combination of backscatter intensity, texture features, and some ancillary information applying different numerical models for classification and segmentation. For example, Aldenhoff et al. [13], based on backscatter intensities in co- and cross-polarization and autocorrelation as a texture feature, used neural networks to achieve the mapping between image features and ice/water classification. Chen et al. [14] used extended-maximum operator, morphological image processing, and geometrical features to identify individual MYI floes from SAR images. However, these studies only considered the differences between the spatial characteristics of different sea ice types. To develop deep learning models for sea ice classification or concentration prediction, Malmgren-Hansen et al. [15] fused Sentinel-1 SAR images with Advanced Microwave Scanning Radiometer 2 (AMSR2) data to produce the AI4Arctic/ASIP Sea Ice Dataset and used it to train a Convolutional Neural Network (CNN) architecture for sea ice concentration prediction. Khaleghian et al. [16] combined near-simultaneous SAR images from Sentinel-1 and optical data from Sentinel-2 to train a CNN sea ice classification model. Both studies have focused on the use of complimentary information to improve the model performance.

For this study, we used SAR image dataset which is available at `http://ieee-dataport.org/open-access/si-stsar-7`. Following section will briefly discussed details of dataset and how author of dataset created the dataset.

2.1 Dataset details

In this study, spatiotemporal dataset of Sentinel-1 SAR, named SI-STSAR-7 is used [17]. Dataset was constructed using the GRD products of Sentinel-1 with HH and HV polarization modes which are obtained over Hudson Bay in winter. The time span of the SAR data was set to two freeze-up periods of Hudson Bay, from October 2019 to May 2020 and from October 2020 to April 2021. Seven sea ice types with 164,564 samples are there. Where each sample has size of 32 x 32 x 6. 32 x 32 is size of patch while 6 is time steps (means 6 different images were captured over same location but on different time, here the time gap between two consecutive acquisition is six days

apart).

Stage of Development	Thickness (cm)	Code	Color
Open Water (<1/10 Ice)			
New Ice	<10	1	
Grey Ice	10–15	4	
Grey-White Ice	15–30	5	
Thin First-Year Ice	30–70	7	
Medium First-Year Ice	70–120	1.	
Thick First-Year Ice	>120	4.	

Figure 2.1: 7 sea ice type

In this way, dataset have spatio-temporal information of sea ice. Dataset that we used in our study is available at IEEE data portal: http://ieee-dataport.org/open-access/si-stsar-7. In this study, only 7 types of ice are considered out of 15. 7 types of ice that we consider in our study is shown in Figure-1.

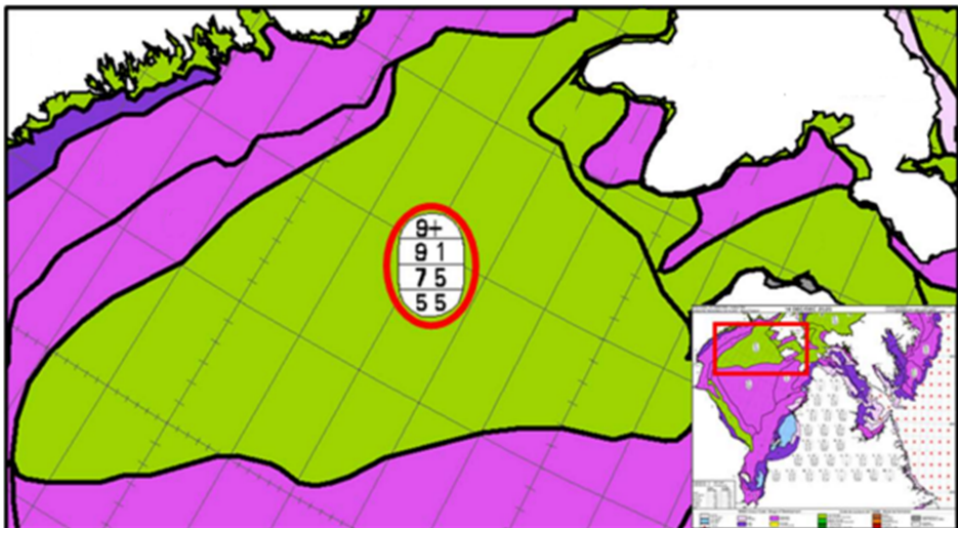

Figure 2.2: Sea ice chart provided by CIS(canadian ice service) [source : CIS official website]

2.2 Dataset construction

Authors of Dataset created dataset using following technique. Firstly Sea ice chart of Hudson bay is acquired by them. Every ice chart has following structure.

CIS gives each ice to specific code and specific color so that we can easily identify which type of ice is available in particular region by seeing the ice chart. Each region is separated from each

other by thick black border line as show in Figure-1 and contain ellipse which gives details about availability of ice type, total concentration of ice and floe(sheet of floating ice) size. In addition to this, each region is colored with specific color wherein, the filled color is dependent upon which type of ice concentration is higher on that region. In Figure-2.2, ellipse contain 4 rows wherein, first row represent total ice concentration of ice in area. Third row contains which types of ice are available in that region while second row contains proportion of ice which are listed in third row. Lastly, Forth row contains size of floe. CIS gave code to each type of ice and even to floe size, so, respective code is written down in Figure-2.3.

Figure 2.3: ASF vertex

Once ice chart of particular time is acquired, then region having higher concentration of particular ice become area of interest because sample that will be taken out from that region have higher chance of having the class that has higher concentration in that particular region. Once area is selected then image of that particular region need to be downloaded from Open Access Hub of sentinel-1.

Sentinel-1 is constellation of two satellite, named sentinel-1A and sentinel-1B. Both are polar orbiting satellite which are having phase difference of 180°. Near polar region it provide repetition period of 6 days while near equator region it provide repetition period of 12 days. Here Area of interest for our studies lies at polar region henceforth it would be possible to have images of same Geo-location that are six days apart. But those captured images may not exactly coincide on each other because one is captured in sentinel-1A and other is captured using sentinel-1B so in that case, intersection of that captured images need to be extracted and then analysed. As shown in Figure-2.3, orange box covered the area that were captured by sentinel-1A while blue box covered the area that were captured by sentinel-1B. It is clear from the figure that both captured images does not cover the same area, therefore intersection of image is required.

Once, the image is intersected, then it converted into 32×32 chips. And each chip get labeled

with particular ice type. Labeling is decided based upon last acquisition which means during last acquisition which ice type dominate in that region. if some type-x is dominated during last acquisition then entire stack of size $32 \times 32 \times 6$ is labeled with type-x.

Chapter 3

Implementation

3.1 Pre-processing

The authors have described pre-processing steps that have been performed on images. It include 8 steps. Significance of each step is discussed below and sequence of pre-processing steps is shown in Figure-3.1.

3.1.1 Applying Orbit file

The orbit file provides an accurate position of SAR image. Usually the satellite orbit is detected by many sensors, like mounted gyro, GPS, and also ground observations. To calculate out the precise orbit data takes time, so precise orbit data are not included in many satellite products. Hence, precise orbit file need to be applied separately. Precise orbit file will help to improve the geo-location. There are two types of orbit file mainly supported in sentinel-1 mission.

- Restitude orbit file :- available more quickly, but are not as exact as the precise ones.
- Precise orbit file :- The precise orbits are available within 20 days at latest, but this can vary.

`https://asf.alaska.edu/data-sets/sar-data-sets/sentinel-1/sentinel-1-data-and-imagery/`

3.1.2 Border noise removal

Sentinel-1 images were impacted by the so-called border noise effect. These noise i.e. non–zero values, appear as a thin strip along the borders of both range and azimuth directions. They are thought to be caused by processor failure in documenting the invalid sensing areas[18]. Such noise is removed in this step. Figure-3.2 is border noise removed product.

3.1.3 Thermal noise removal

Thermal noise in SAR imagery is caused by microscopic motion of electron due to temperature and microwave interaction with atmosphere. In this step thermal noise is removed. Figure-3.3 is thermal noise removed product.

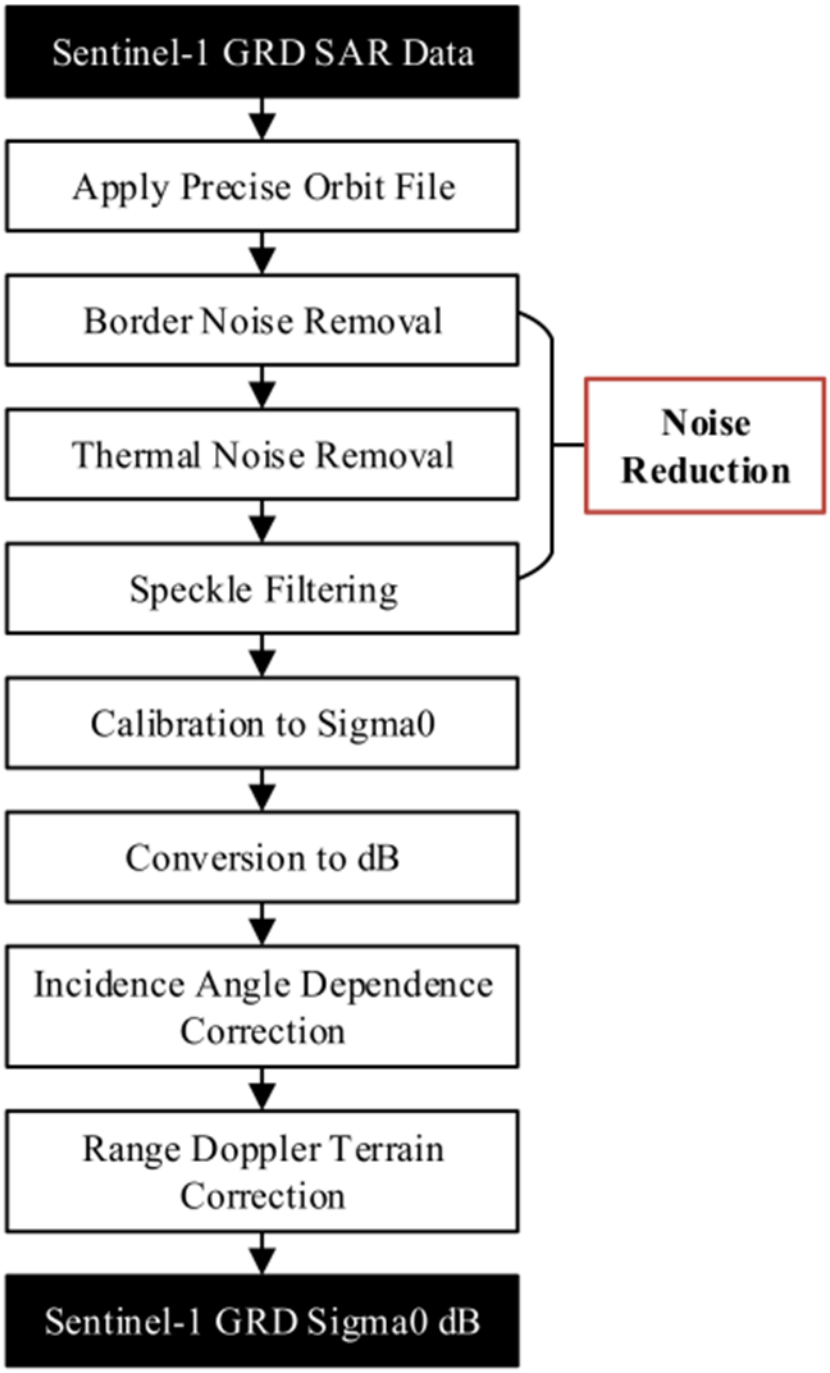

Figure 3.1: Pre-processing steps of SAR image [source:From [1]]

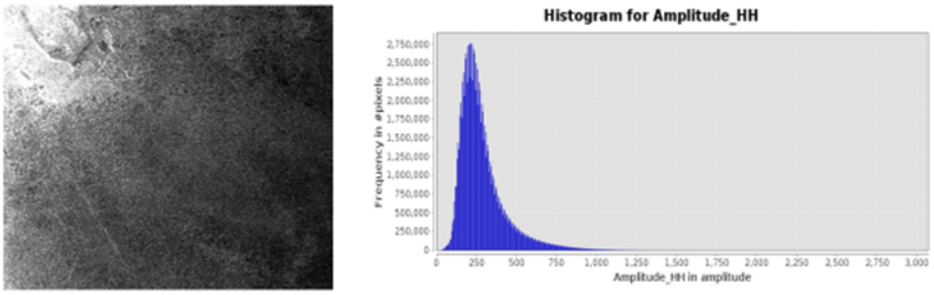

Figure 3.2: Border noise removed Product and its corresponding histogram

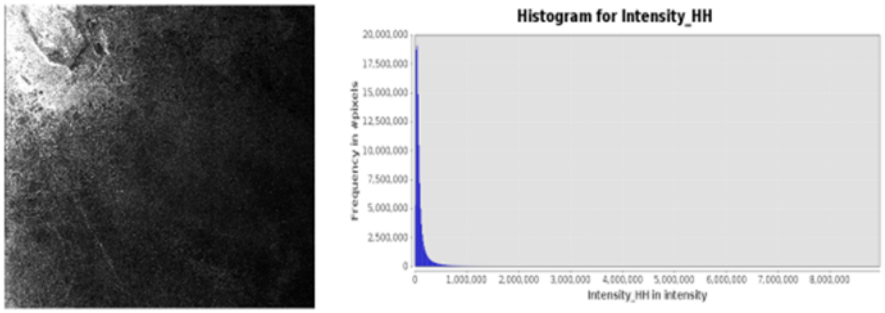

Figure 3.3: Thermal noise removed Product and its corresponding histogram

3.1.4 Speckle noise removal

Speckle noise has granular pattern. It is salt-pepper type of noise that is mainly found in the satellite images. SAR images are formed by coherent interaction of the transmitted microwave with terrain. This coherent interaction causes random constructive and destructive interference resulting in salt pepper noise throughout the image. We used Lee-sigma filter to reduce speckle noise. The parameters of kernel size is 7 x 7, target window size is 3 x 3 and sigma is 0.9. Figure-3.4 is speckle noise removed product.

3.1.5 Calibration to σ^0

Echo (signal received by sensor) that are normalized as per unit area on the ground received from the distributed target is called σ^0. Figure-3.5 is σ_0 calibrated product.

3.1.6 Conversion to dB

In this step, we convert intensity value in dB. Figure-3.6 is dB converted product.

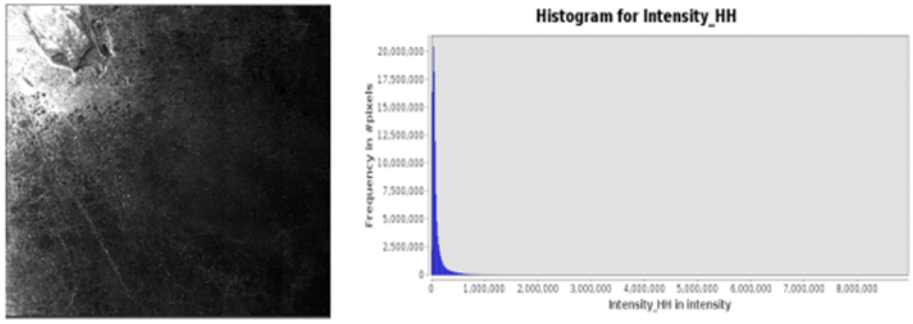

Figure 3.4: Speckle noise removed product and its corresponding histogram

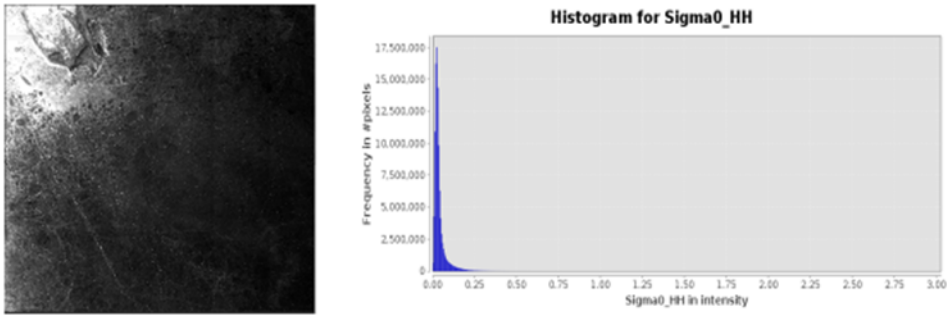

Figure 3.5: σ_0 calibrated product and its corresponding histogram

3.1.7 incident angle dependence correction

Incidence angle dependence correction deals with removing dependence of sigma0 on incidence angle. As Incidence angle increases, σ^0 decreases. So, to remove that effect, below equation is used.

$$\sigma^0_N = \sigma^0 + c \times (\theta - \theta_0)$$

Where σ^0, is original σ^0 value before correction, θ is actual angle while θ_0 is value of angle at centre of swath. C is constant which value is -0.24db/° for HH polarization while -16db/° for HV polarization. Value of c is calculated using regression. A lot of research results show that the linear regression model well describes the dependence of σ_0 in dB scale on θ. Figure-3.7 is incident angle dependence corrected product.

3.1.8 Range-Doppler terrain correction

There is an impact of geometric distortion on the SAR images due to characteristic of sensor, platform and object. Range Doppler terrain correction reduces the geometrical distortions present in SAR image. Figure-3.8 is range-doppler terrain corrected product.

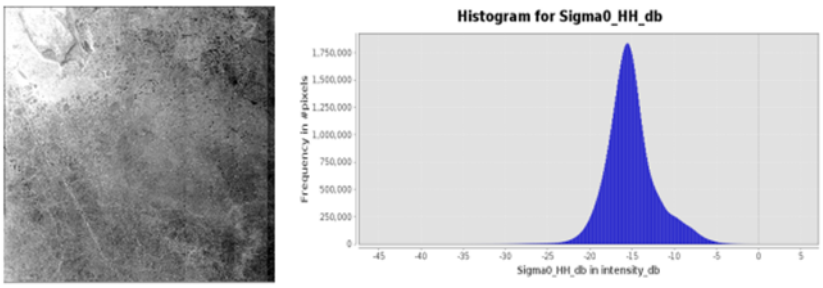

Figure 3.6: dB Converted Product and its corresponding histogram

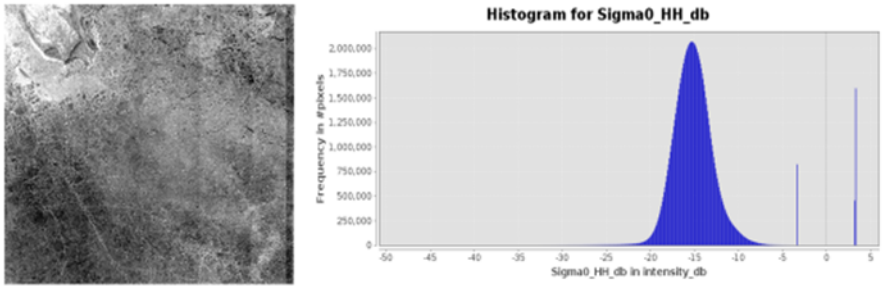

Figure 3.7: incident angle dependence corrected product and its corresponding histogram

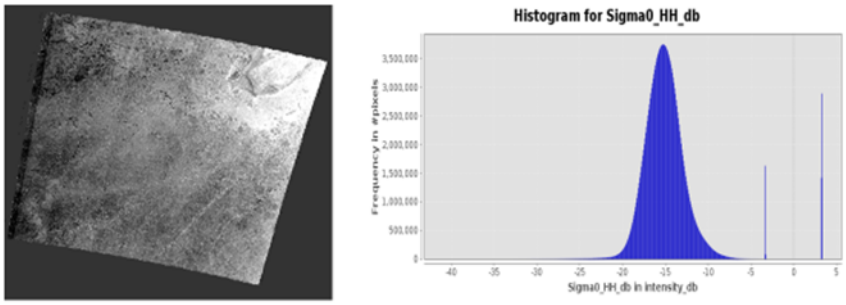

Figure 3.8: Range-doppler-terrain corrected product and its corresponding histogram

3.2 Applied models

In this study, we developed two models. First is Resnet and second is convLSTM. Both models are developed in Python(version 3.9) using Pytorch and TensorFlow packages respectively.

Traning was conducted on a computer with the following specification: Redhat linux operating system, NVIDIA TITAN RTX GPU having 40GB memory with MPR architecture.

3.2.1 Resnet

A residual neural-network (ResNet) is an artificial neural network. Very deep feed-forward neural-network with hundreds of layers, it is deeper than previous neural networks. Generally, when we increase depth of network at that time accuracy of modeled task is also getting increased so for that reason it will become more and more common to have deeper architecture to gain better result. However, it will not hold true as there is a problem of vanishing gradient. As shown in Figure-3.9, theoretically accuracy would increase as we increase depth of model due to increase in capacity of model however, practically it will not increase after certain depth because of vanishing gradient phenomenon.

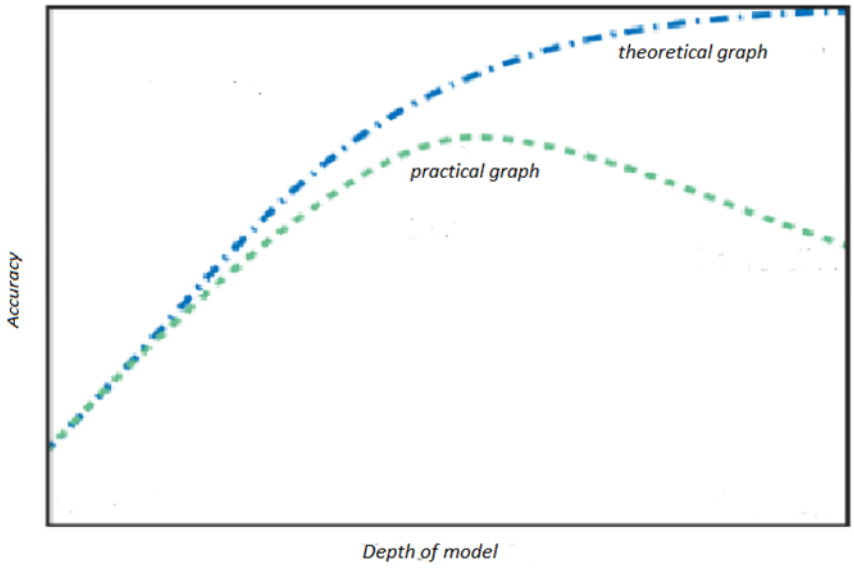

Figure 3.9: practical and theoretical result of impact of depth on model accuracy [source : medium]

Theoretically, accuracy should increase with an increase in depth of model but as depth of model increase phenomenon called vanishing gradient also become significant and thus there will be no effective learning will be carried out at initial layers of graph. As all neural network layers are learned its weights through the procedure called gradient descent. In gradient descent, weights are changed in such a way that it always reduce model's loss function. In each neurons of neural network layers, we first have linear transformation(multiplying input with weights) and then non-linearity. Non-linearity can be in form of Sigmoid and ReLU. And as output of one layer would be input to another layer so if derivation of loss with respect to inner most parameter is required then in that case, that derivation term will have derivation of non-linearity term for all layers that were exceeding it. Derivation of non-linearity (such as sigmoid) have value in between $-\frac{1}{2}$ to $+\frac{1}{2}$.

As number of layers increases, number of derivative terms in inner most parameter derivation also will increase and this will in turn nullify the effect of learn and thus this phenomenon called vanishing gradient descent. Because if we have ten such layers then first layers parameters will

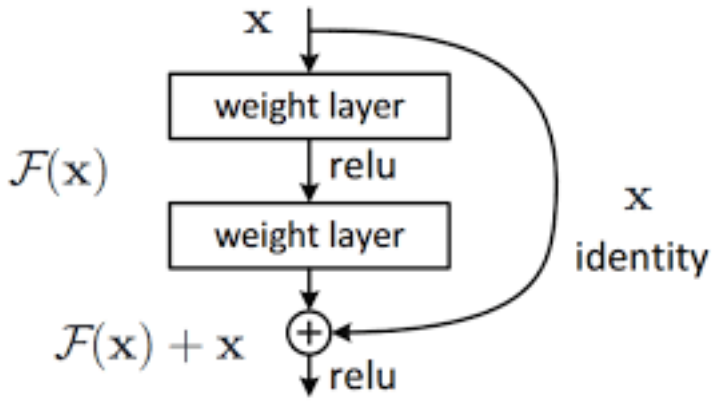

Figure 3.10: skip connnection

have nine derivative term of exceeding layers' non-linearity that would count to $\frac{1}{512}$ which is equal to 0.0019 which is still bearable but if number of layers will increase to 20 then that count will be accounted for 1.90e-6 which can easily nullify the learning of starting layers.

To diminish the effect of vanishing gradient problem skip connection is used. In which input of layer is added or appended with layers' output as shown in Figure-3.10. Input to the layer "x" is added with output to that layer (that is applying linear transformation and non-linearity). So doing this will help gradient to pass through that addition part to previous layer and in this way vanishing gradient phenomenon is diminished.

3.2.2 convLSTM

ConvLSTM is recurrent neural network. Moreover, it is designed to work for images instead of vector(one dimensional matrix). To put it simply it is just a 2D version of LSTM. Same as LSTM it has input gate, output gate and forget gate. In the LSTM, all the gates are comprise of multiplying input and hidden state with weight matrix and then adding them together while in ConvLSTM everything will be remained same, instead multiplying input and hidden state with weight matrix would change with convolution operation . And this make sense as ConvLSTM is used with images and Convolution is operation which extract meaningful information out of image. Thus "Conv" is in the name of ConvLSTM.

As shown in Figure-3.11, Firstly convolution will be applied to input images as well as on hidden state, and output of both of the operation will be used to further calculate each gate output (here input gate, output gate and forget gate) and new information. In ConvLSTM, cell state is there, in which relevant information is going to get stored so that latter that information will be used to make accurate prediction. And which information will get stored is decided by forget get and input gate. Output of forget gate is multiplied (element wise) with cell state, here forget gate output would be in between 0 to 1 because sigmoid function sits after applying convolutional layer. If output of forget gate is zero then it means to wipe-out the cell-state information and if output of forget gate is one then it means to keep cell-state information and in between value have considerate significance on remembering as well as forgetting information. Later , output of input gate is multiplied (element wise) with new information and then it will be added up to cell state.

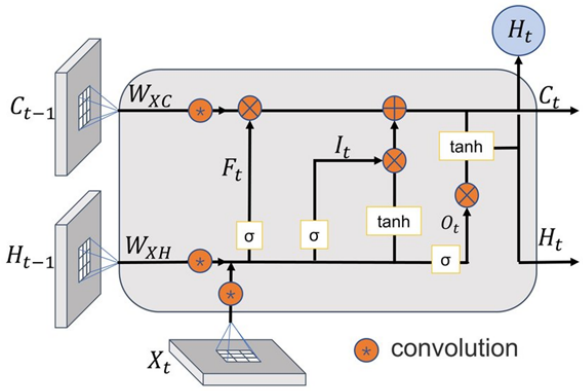

Figure 3.11: ConvLSTM cell [sources:wikipedia]

Output gate's output is multiplied with cell state later to produce new hidden state and this way architecture is designed in such a way so that it can learn to remember significant information for prediction.

Chapter 4

Experiments and Results

4.1 Experiments

4.1.1 Hyperparameter tunning

Resnet

For hyperparameter tunning we use the parameters that are listed in Table-1.

LR(learning Rate)	0.01 , 0.001 , 0.0001 , 0.00001
Batch Size	64 , 128 , 256 , 512
Optimizer	SGD , RMSprop , Adam

Table 4.1: set of hyperparameter used in Resnet training

In Figure - D.4,D.5,D.6(in appendix-D), we hand picked some of the best performing set of parameters graph out of all. And after carefully analysing all graph, we picked 0.001 (LR), RMSprop(optimizer) and 256(batch-size) as our hyperparameter for training the model.

convLSTM

For hyperparameter tunning we use the parameters that are listed in Table-2.

LR(learning Rate)	0.01 , 0.001 , 0.0001 , 0.00001
Batch Size	64 , 128 , 256 , 512
Optimizer	SGD , RMSprop , Adam

Table 4.2: set of hyperparameter used in convLSTM training

In Figure - D.1,D.2,D.3(in appendix-D), we hand picked some of the best performing set of parameters graph out of all. And after carefully analysing all graph, we picked 0.01 (LR), RMSprop(optimizer) and 128(batch-size) as our hyperparameter for training the model.

4.2 Results

4.2.1 Resnet

Figure-4.1 depict the accuracy and loss per epoch achieved during training. Resnet-34 is used in training and it was trained for 200 epochs where it attained highest testing accuracy of 94.48%. Training accuracy would reach up to 99.86%. For training we took 0.001 learning rate with batch size of 256 and optimizer was RMSprop.

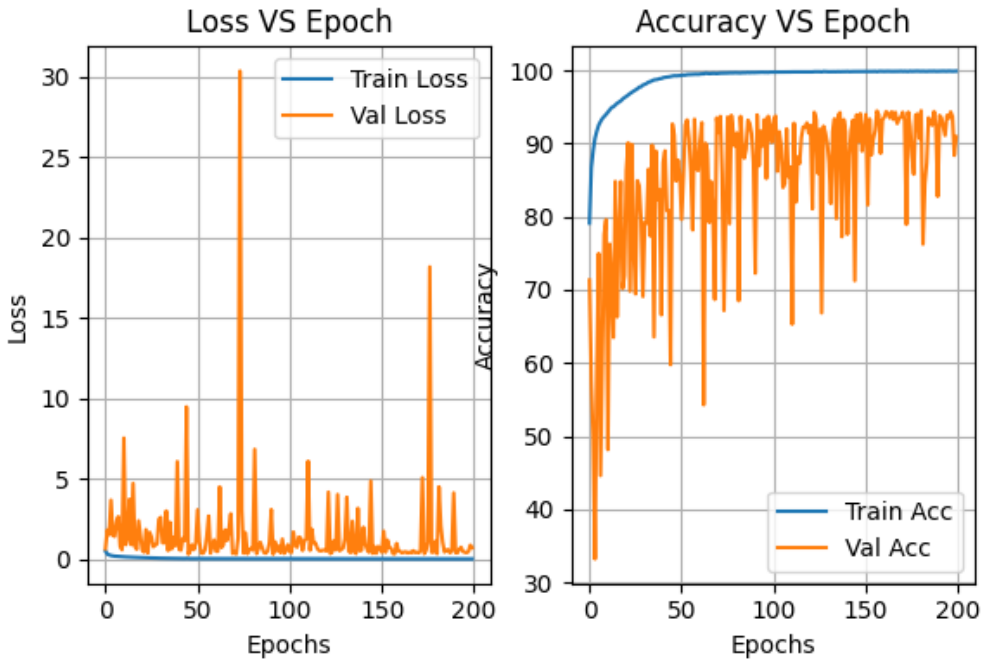

Figure 4.1: Resnet loss and accuracy graph

Train,Test,validtion split	0.8,0.1,0.1
LR(learning rate)	0.001
Optimizer	RMSprop
Batch Size	256
Training accuracy	99.86
Testing accuracy	94.48
Training loss	0.0043
Testing loss	0.33
Time per epoch	2 min

Table 4.3: Result of Resnet

4.2.2 convLSTM

Figure-4.2, depict the accuracy and loss per epoch achieved during training. ConvLSTM is used in training and it was trained for 200 during that it achieved training accuracy of 99.99% while getting 96.33% testing accuracy.

For training, we took 0.01 learning rate with batch size was 128 and rmsprop optimizer.

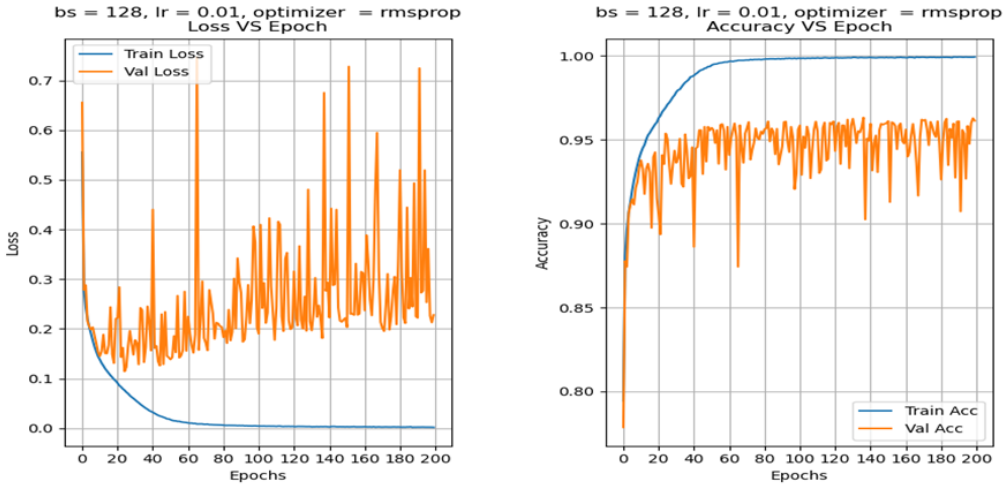

Figure 4.2: ConvLSTM loss and accuracy graph

Train,Test,validtion split	0.8,0.1,0.1
LR(learning rate)	0.01
Optimizer	RMSprop
Batch Size	128
Training accuracy	99.99
Testing accuracy	96.33
Training loss	0.01
Time per epoch	3 min

Table 4.4: Result of ConvLSTM

k-fold cross validation is applied on convLSTM and in that average accuracy of 96.33 was achieved. Corresponding graphs of that is shown in appendix-D.

4.3 validating Result

For validating our Results we downloaded data from ASF-vertex. We downloaded products of year 2018 and 2022, because in training 2019 and 2020 year data is used. For 2018, we downloaded 6

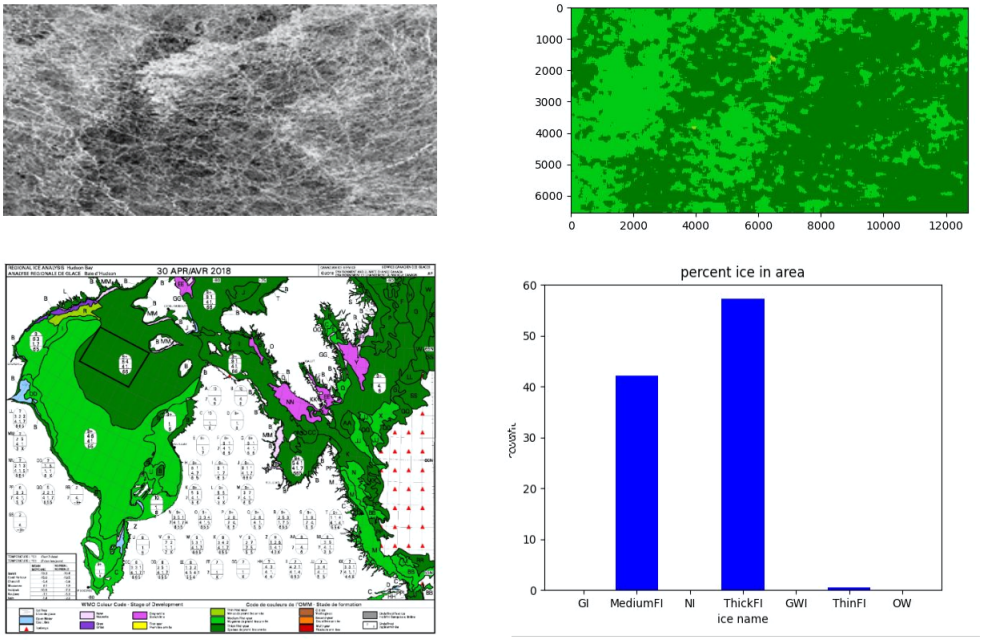

Figure 4.3: Result of Product 2018 and its corresponding ice-chart

product from 22-03 to 27-04(with time gap of 6 days). For 2022, we downloaded 2 product from 12-04 to 24-04(with time gap of 12 days). Once, product is downloaded we first did 8 pre-processing steps on it and later convert that output into 32 $\times$ 32 chips. Lastly, we fed that chips into our model(convLSTM) and get the result and from that result we generate one image and bar-chart. Latter we compared our results with ice-chart(provided by CIS) for validation. Results of the products and its corresponding ice-chart are depicted in Figure-4.3 and Figure-4.4.

In result of 2018, dark green color has been reproduced for 60% of total chips. And ice-chart also shows 60% probability for this region.

In result of 2022, Our model miss classify for almost all region. The reason could be the lack of data. Model has been trained on 6 images having gap of 6 days however here we our only giving 2 images which are having gap of 12 days.

4.4 Entire process

First, we have built the model using training-dataset that is readily available in our case at IEEE dataportal. We have selected two models, first one is convLSTM while, second one is Resnet. We train both of the model on training dataset and store the weight of trained model. Later, at inference time we load that weights of model and using that trained weight for inference. Figure-4.5, depict flowchart of training procedure of model.

Once model is trained then products(SAR images) need to be downloaded and pre-processed(steps as mentioned in Figure-4). After that it need to be converted into a form which could be fed into the model and inference can be obtained. After getting output of images, it is represented in following

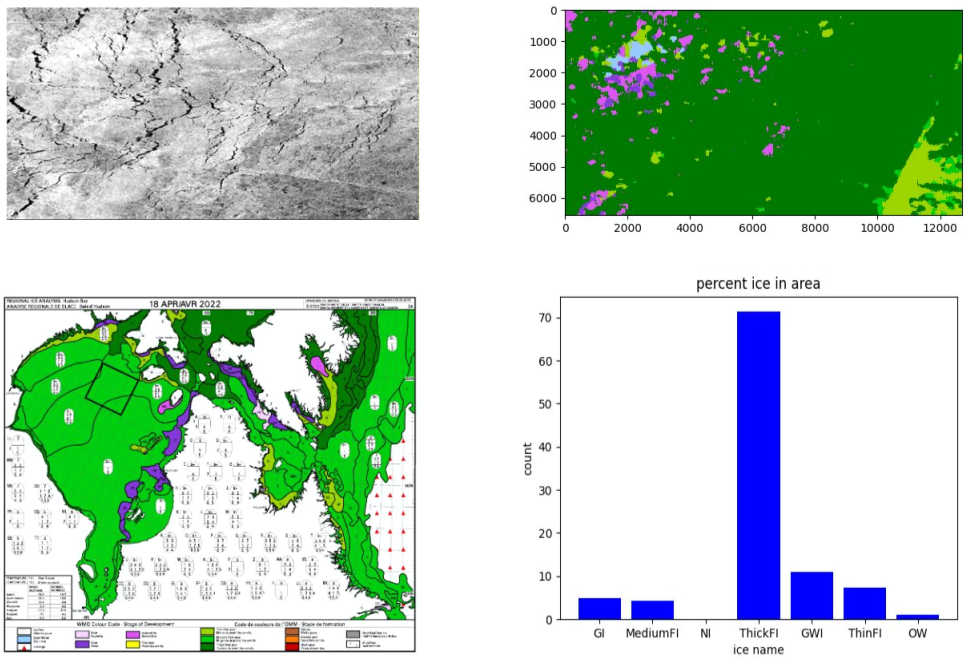

Figure 4.4: Result of Product 2022 and its corresponding ice-chart

forms: 1. bar graph, representing the proportion of each ice, 2. image which shows area that are having particular ice type in that region.

To, automate this whole procedure, First image need to be downloaded from the internet. So for that shell script was written which took care of whole downloading process. ASF(Alaska satellite facility) vertex is the platform from where images were downloaded. Shell script will take some input such as starting date, ending date, relative orbit number and frame. Based on given input, it will download images from the vertex. Once product is downloaded, next task is to apply pre-processing steps onto downloaded image. For pre-processing, we use GPT(Graph Processing Tool) utility of SNAP toolkit. GPT will take graph as input (which was written in .xml file) and generate the output. All steps except incidence angle dependent correction (Figure-3.1, step no. 7) is done through SNAP toolbox. For incidence angle dependent correction, another python script was written. Lastly, all these pre-processing steps were combined into one python script file and then fire 6 processing on 6 different product and in this way parallelism was achieved. And time was reduced from 14 min to process all the product to 4 min.

Once, pre-processing steps were completed, images need to be converted into 32 x 32 chips and that chips need to be fed into the model to get the output. After output is generated, we need to convert it into proper format like will need to create bar graph which will represent proportion of ice types and image which illustrates which ice is available in which region. Figure-4.6's flow-chart depict how all process have been carried out.

Lastly, we have integrated our product with pre-built GUI(graphical user interface). That GUI takes same input as our command line scripts. And at the end, shows result on GUI panel. Figure-21 contains GUI interface where appropriate input need to be provided to get the output.

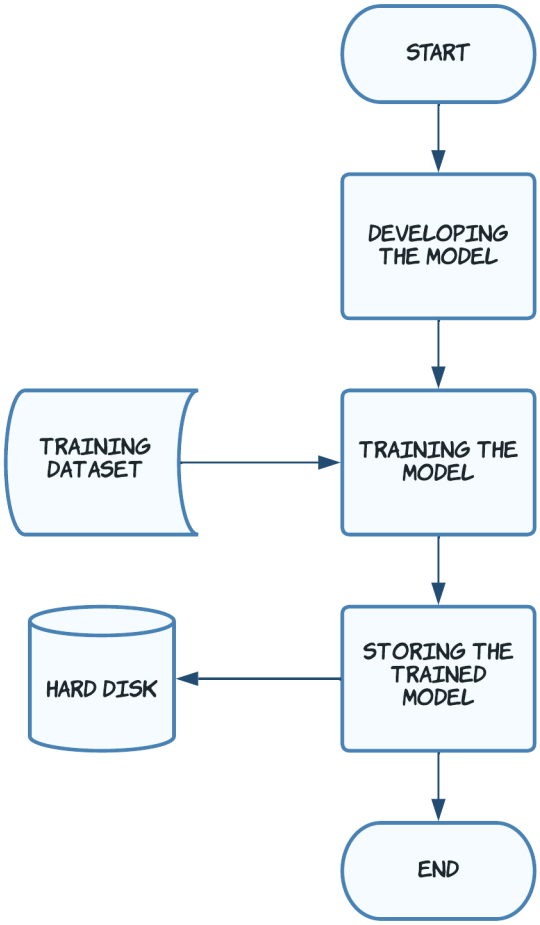

Figure 4.5: Training procedure workflow

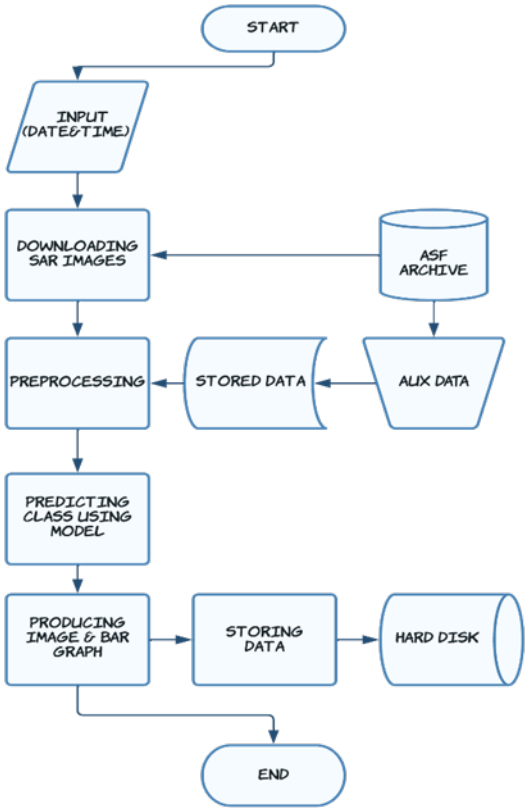

Figure 4.6: Flowchart of entire process

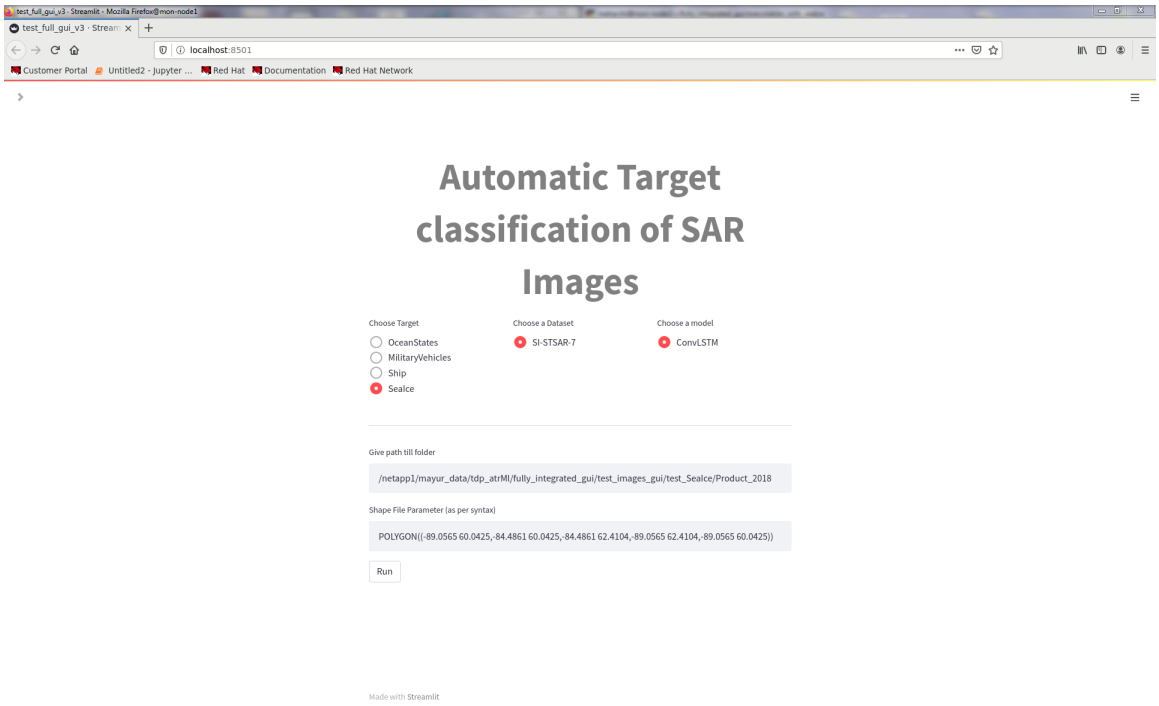

Figure 4.7: GUI Interface

Chapter 5

Conclusion and Future work

In this project, we have trained two models named Resnet and ConvLSTM. In both the models I got accuracy of 94% and 97% respectively. After training the model, my task is to download SAR images from the internet and do all pre-processing steps.Then apply my model on new downloaded images after converting it to 32 $\times$ 32 chips. For downloading SAR images, I have written linux script which will carry out downloading of product from internet(ASF-vertex). For pre-processing, I have used SNAP toolkit which is provided by ESA (European Space Agency). Once, pre-processing is completed, we fed this pre-processed input to model and model will in-turn give us prediction. Using that predictions we built two type of output which we have mentioned in Results section .

Accuracy that we achieved using above mentioned models is quite good. Accuracy may increase if we use visual transformer. In future, I will apply visual transformer on my task to see whether accuracy will increase further or not.

Bibliography

[1] Clausi, D.A. An Analysis of Co-occurrence Texture Statistics as a Function of Grey Level Quantization. Can. J. Remote Sens. 2002,28, 45–62

[2] Ressel, R.; Frost, A.; Lehner, S. A Neural Network-Based Classification for Sea Ice Types on X-Band SAR Images. IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. 2015, 8, 1–9. [CrossRef]

[3] Ressel, R.; Singha, S.; Lehner, S.; Rösel, A.; Spreen, G. Investigation into Different Polarimetric Features for Sea Ice Classification Using X-Band Synthetic Aperture Radar. IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. 2016, 9, 3131–3143

[4] Clausi, D.A. An Analysis of Co-occurrence Texture Statistics as a Function of Grey Level Quantization. Can. J. Remote Sens. 2002, 28, 45–62.

[5] Deng, H.; Clausi, D.A. Unsupervised Segmentation of Synthetic Aperture Radar Sea Ice Imagery Using a Novel Markov Random Field Model. IEEE Trans. Geosci. Remote 2005, 43, 528–538.

[6] Ochilov, S.; Clausi, D.A. Operational SAR Sea-Ice Image Classification. IEEE Trans. Geosci. Remote 2012, 50, 4397–4408.

[7] Ressel, R.; Frost, A.; Lehner, S. A Neural Network-Based Classification for Sea Ice Types on X-Band SAR Images. IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. 2015, 8, 1–9.

[8] Ressel, R.; Singha, S.; Lehner, S.; Rösel, A.; Spreen, G. Investigation into Different Polarimetric Features for Sea Ice Classification Using X-Band Synthetic Aperture Radar. IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. 2016, 9, 3131–3143.

[9] Liu, H.; Guo, H.; Zhang, L. SVM-Based Sea Ice Classification Using Textural Features and Concentration From RADARSAT-2 Dual-Pol ScanSAR Data. IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. 2015, 8, 1601–1613.

[10] Zhang, L.; Liu, H.; Gu, X.; Guo, H.; Chen, J.; Liu, G. Sea Ice Classification Using TerraSAR-X ScanSAR Data With Removal of Scalloping and Interscan Banding. IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. 2019, 12, 589–598.

[11] Yang, K.-S.; Kiang, J.-F. Comparison of Algorithms and Input Vectors for Sea-Ice Classification with L-Band PolSAR Data. Prog. Electromagn. Res. B 2019, 84, 1–21.

[12] Boulze, H.; Korosov, A.; Brajard, J. Classification of Sea Ice Types in Sentinel-1 SAR Data Using Convolutional Neural Networks. Remote Sens. 2020, 12, 2165.

[13] Aldenhoff, W.; Heuzé, C.; Eriksson, L.E.B. Comparison of Ice/Water Classification in Fram Strait from C- and L-band SAR imagery. Ann. Glaciol. 2018, 59, 112–123.

[14] Chen, S.; Shokr, M.; Li, X.; Ye, Y.; Zhang, Z.; Hui, F.; Cheng, X. MYI Floes Identification Based on the Texture and Shape Feature from Dual-Polarized Sentinel-1 Imagery. Remote Sens. 2020, 12, 3221.

[15] Malmgren-Hansen, D.; Pedersen, L.T.; Nielsen, A.A.; Kreiner, M.B.; Saldo, R.; Skriver, H.; Lavelle, J.; Buus-Hinkler, J.; Krane, K.H. A Convolutional Neural Network Architecture for Sentinel-1 and AMSR2 Data Fusion. IEEE Trans. Geosci. Remote 2021, 59, 1890–1902.

[16] Khaleghian, S.; Ullah, H.; Kræmer, T.; Hughes, N.; Eltoft, T.; Marinoni, A. Sea Ice Classification of SAR Imagery Based on Convolution Neural Networks. Remote Sens. 2021, 13, 1734.

[17] Wei Song, Wen Gao, Qi He, Antonio Liotta, Weiqi Guo, June 18, 2021, "SI-STSAR-7", IEEE Dataport, doi: https://dx.doi.org/10.21227/d6kp-s174

[18] Luo Y, Flett D. Sentinel-1 Data Border Noise Removal and Seamless Synthetic Aperture Radar Mosaic Generation. Proceedings. 2018; 2(7):330. https://doi.org/10.3390/ecrs-2-05143

Appendix A

Parameter learning

Start with very simple example, suppose Figure-A.1(A) scatter plot is given and task is to fit line which best represent the following plot

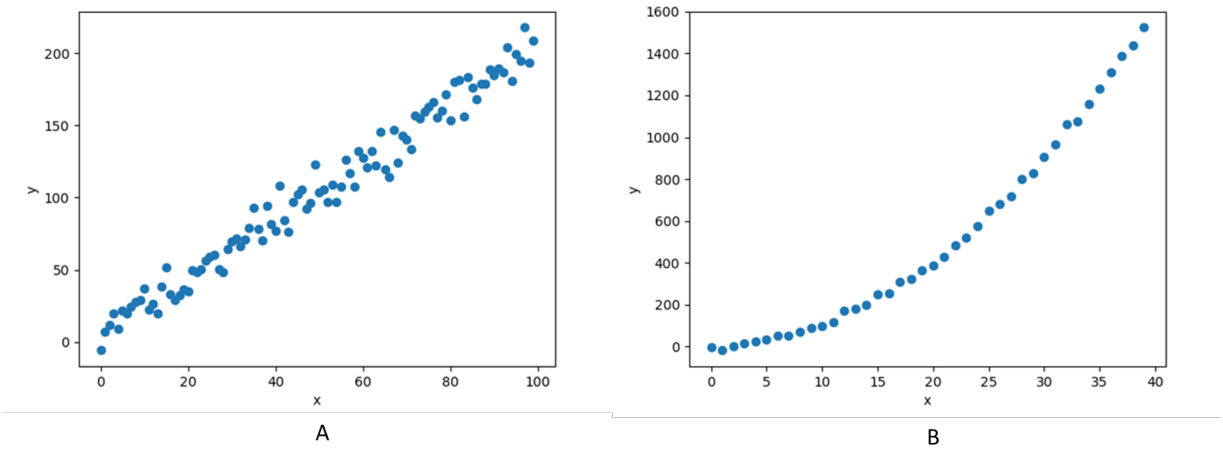

Figure A.1: Caption

In order to fit line, we required to estimate, quantity 'y' is depended upon which other quantity. From Figure-A.1(A), it is clear that 'y' is solely depended upon 'x' and there is a linear co-relation between two quantity hence our estimate should be

$$\hat{y} = \theta_1 \times X + \theta_2$$

other than linear co-relation, we have to estimate equation differently. In Figure-A.2, quantity 'y' is depended upon 'x' but not linearly which is clear after seeing the scatter graph. It looks like, y is in linear co-relation with X^2. Therefore, in that case the estimate for line should be $\hat{y} = \theta_1 \times$ X + θ_2.

For parameters estimation, we consider Figure-A.1(A) case. So, now the task is to estimate the parameters θ_1, θ_2 in equation $\hat{y} = \theta_1 \times$ X + θ_2. But before that, it is required to measure how good our estimate $\hat{y}$ is. So, for that reason, some way has to be found, in order to measure

goodness of our estimate.Basically, L2-norm is used as measure for goodness(mostly for regression task). $\|y–\hat{y}\|_2 = (y–\hat{y})^2$. Now question is why L2-norm is used and why are not others? There are some probability theory and reasoning which backs the idea of L_2-norm but there is nothing special about L2-norm. Other loss like L1 can be used as well.

Measure of the goodness for our estimate is achieved. L_2–norm gives measure about how bad our model is, hence, from now on wards we call it as loss and in literature loss term is prevalent. Lower the loss better the estimate would and wise versa.

$$Loss = (y - \hat{y})^2$$

Now, everything is achieved including estimate and way to measure goodness of our estimate. But, still one thing is missing and that is how to find the best parameter so that the loss became lowest or at-least could be reached to local minimum if there are any.

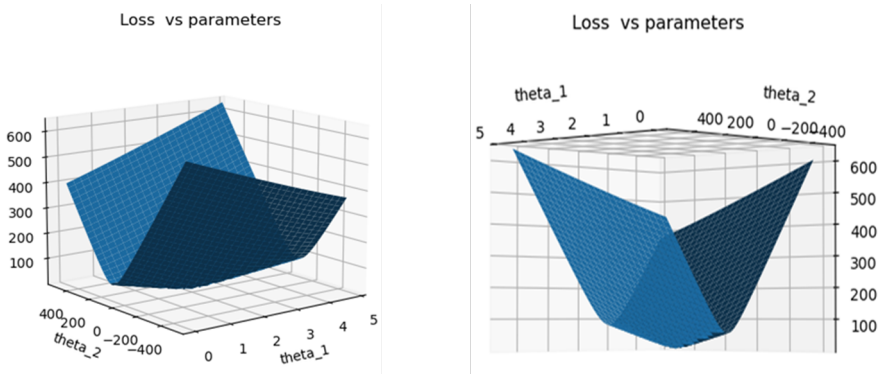

Figure A.2: loss vs parameters plot

Figure-A.2 is 'Loss vs (θ_1 & θ_2)'. It has convex shape. So, somehow parameters can be found for which our loss is lowest then our task is completed. But how to find such parameters for which the loss is lowest?

This problem is totally opposite of hill climbing problem. In hill climbing problem, climber want to move in direction which is steepest ascent so in lesser time, climber can climb the mountain. In same way here, we want to climb down in such a way so that we can reach to bottom as fast as possible. However, how the direction of steepest ascent or descent can be found?

According to vector calculus, direction of gradient is always in the direction of steepest ascent. So opposite direction of that would be the direction of steepest descent. Now obvious question will be arose in our mind, why the direction of gradient is in the direction of steepest ascent?

Proof.

Let's suppose we are at point (x, y) and we want to move in the direction, which is steepest descent. So suppose we take one arbitrary vector $\bar{v}$ of unit length and as we know that, directional derivative will give us the rate at which value of function would change if we move in that direction. We want to take step in the direction which is steepest ascent. So our vector $\bar{v}$ should be the one in which maximum rate of change in function value is there. So mathematically we write,

$$\underset{x}{\text{argmax}} \nabla_{\bar{v} f(x) = \nabla f(x) \cdot \bar{v}} \tag{A.1}$$

$$\underset{x}{\text{argmax}} \nabla_{\bar{v} f(x) = \|\nabla f(x)\| \cdot \|\bar{v}\| \cdot \cos\theta} \tag{A.2}$$

This this quantity will only became maximum, when $\cos\theta = 1$, and that would happen only if angle between both vector ∇f(x) and $\bar{v}$ is zero, which means that both are in same direction. Hence, the direction of gradient is in the direction of steepest ascent is proved. Therefore, if we want to climb-down the hill rapidly then the move in the opposite direction of gradient should be taken. Now, we have all tools to solve the problem of finding best parameters.

The problem is to find the optimal parameter for $\hat{y} = \theta_1 \times \text{X} + \theta_2$. Here, we can initialize θ_1, θ_2 with random value. And then change the value of θ_1, θ_2 such that loss at every step loss will become lesser and lesser than before. So, we change θ_1, θ_2 in opposite direction of gradient of Loss (Which is here in our case is L_2 Norm).

```
Until θ1, θ2 converged:
    θ = θ - α x θ x ∇Loss(y, ŷ)
```

Where 'α' is learning rate. It is required to tune learning rate because if it is too small then θ will never converge and if it is too big then it will overshoot the objective and it will diverge instead of converging. Figure-A.3 shows the both example.

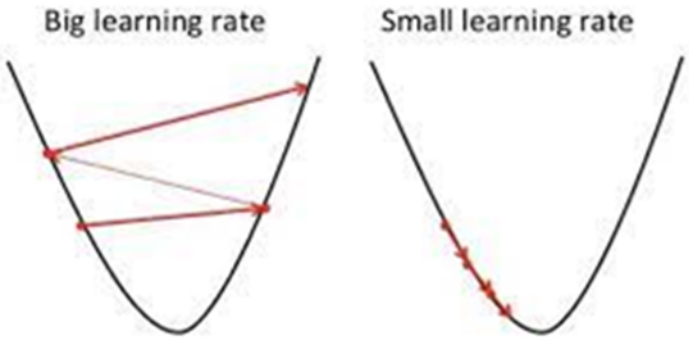

Figure A.3: effect of learning rate on convergence

As we have discussed, we first need to estimate equation for our line. For Figure A.1(A). We estimate line equation as $\hat{y} = \theta_1 \times \text{X} + \theta_1$ while For Figure A.1(B) we estimate line equation as $\hat{y} = \theta_1 \times \text{X}^2 + \theta_1$. But all the time it is not possible to correctly estimate equation for line by seeing the graph only and here number of dependent variable is less but what if it is too high, so in that case we cannot plot a graph. Therefore, the solution for this problem need to be found. Suppose, if we are not able to correctly estimate equation of line then model have high bias(difference between the average prediction of our model and the correct value which we are trying to predict) but low

variance(sensitivity to small fluctuations) and if we overestimate the equation then our model has low bias but high variance. Both the cases are unacceptable. Balance between this two will be required to achieve higher accuracy.

Solution to the problem, is to use more complex model such as neural network. Neural network has following structure. Input is given to input layer. Weighted sum of input is latter transform by non-linearity inside all neural in network. Output of input layer now feed into second layer and we keep on doing this until we have output at the end of layer. Once we have the output then we can calculate loss function and then follow the same procedure as above. Change the parameter in direction where loss is getting reduced. Finding gradient of Loss with respect to learn-able parameter in neural network is bit tedious task, henceforth, we often use software package which do that task for us.

Neural network is not only useful for regression (find relationship between independent and dependent variable or in simple word, find line which best represent our data) but is also used for classification (to distinguished item based on available information to particular category). NN can also be learned from images. Research showed that it would be better to use Convolution in NN rather than using dense connected NN.

As of now numerous convolution neural network architecture exist like LeNET, AlexNET, VGG, Inception, ResNet, and Xception. All of that are discussed in below passages. Common thing in all those architecture is that each have more or less three same type of components. Firstly Convolutional layer with non-linearity, Secondly Pooling layer and lastly fully connected dense layers.

Universal approximation theorem.

if there is a function f(x):R^m -> R^n, then there exits neural network(NN) with hidden layers that approximate f(x).Which means that suppose function represented by neural network is N(x), then $\|f(x) - N(x)\| < \epsilon$

Appendix B

Layers in deep-neural network

Dense Layer.

Dense layers is fully connected neurons as shown in below diagram. Each neurons in layer is connected with previous layer neurons. Each neurons have mainly two part; first, applying weighted average on previous layer output. Then applying non-linearity. There are several drawback of applying dense layers directly on images. Dense layer will use one neuron for each input so as dimension of input image increase then the amount of weights to learn are also rapidly increase. So for that reason dense layers cannot be use straight away on images. First Convolution would be applied on image and then those extracted features would be fed into dense layers for inference

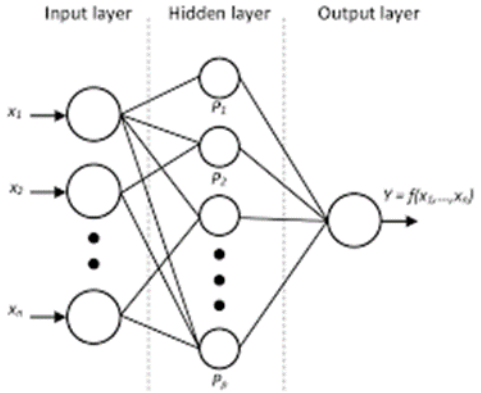

Figure B.1: Dense Layer

Convolutional layers.

Using dense layers directly on image is inefficient. And second problem is that, when given image is got flatten for dense layer then spatial information is lost. To capture spatial information of image, CNN would be better choice to use because of sparse interaction and weight sharing. Sparse interaction means that only nearby pixels of given pixel will be used to calculate output of convolutional layer for that location. And kernel will be moved over all image pixels so for all the patches in image, kernel weights would remain the same that's how in convolutional layer we have weight sharing. In most cases output of convolutional layers will passed through some non-

linearity like ReLU, Sigmoid, and Tanh. Non-linearity is required because without it our model only have linear transformation and with that we could achieve optimal result. Some of the term which frequently used convolutional layer is kernel$_s ize, padding, strideetc.$

- Kernel size means width and height of kernel (also known as filter) that we used in convolution layer.
- Padding means we add redundant information at the border of image for sake of processing.

  Ex: if padding is K then we add K rows and columns at the border of image
- Stride means at what interval kernel will be moved over image.

  Ex: if stride is K then we skip K-1 rows and columns

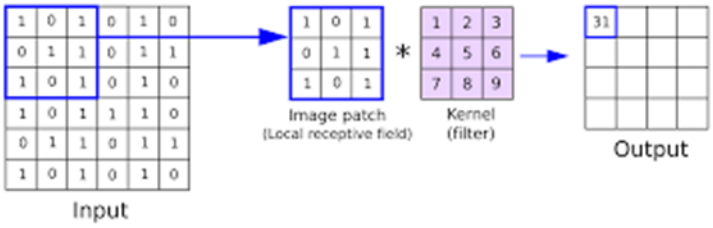

Figure B.2: Convolution operation example

Pooling layers.

The pooling layer replaces the output of the network at certain location by deriving a summary statistic of nearby outputs. This helps in reducing the spatial size of the representation, which decreases the required amount of computation and weights. The pooling operation is processed on every slice of the representation individually.

There are several pooling functions such as the max pooling and average pooling. In max pooling, maximum of given area will be outputted while in average pooling, average of given area will outputted. Size of features space will significantly reduce.

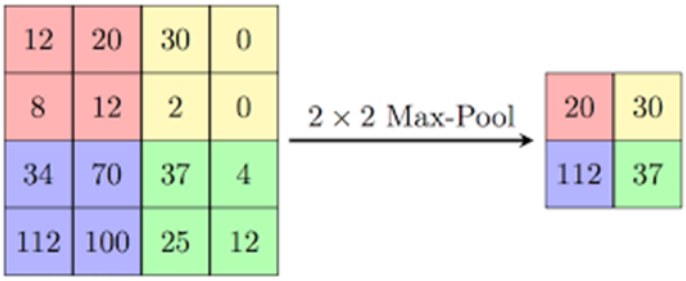

Figure B.3: Pooling operation example

Appendix C

DNN (deep neural network) Architecture detail

LeNET(1998).

- 5 Learnable layers with two convolutional layers while three fully connected dense layer
- Expected image size should be 28 x 28 with one channel.
- Two Convolution layers having 6 and 16 kernels respectively of size 5 x 5.
- Each convolutional layers followed by average pooling layers. Which will reduce the feature size to half. Pooling layer have size of 3 x 3. At last architecture have three fully connected dense layer. Which are having sigmoid as non-linearity

advantage

- Less parameter to learn thus it took less time to converge.
- It took less memory since it has fewer parameter in comparison with other.
- It is Simple & Robust.

disadvantage

- Overfitting in some cases and there is not built-in mechanism to avoid this.
  - So benchmark model is improved by adding dropout layer.
- Fewer number of learn-able parameters means that it has fewer capacity. So for large dataset it may suffer from under fitting

AlexNET(2013).

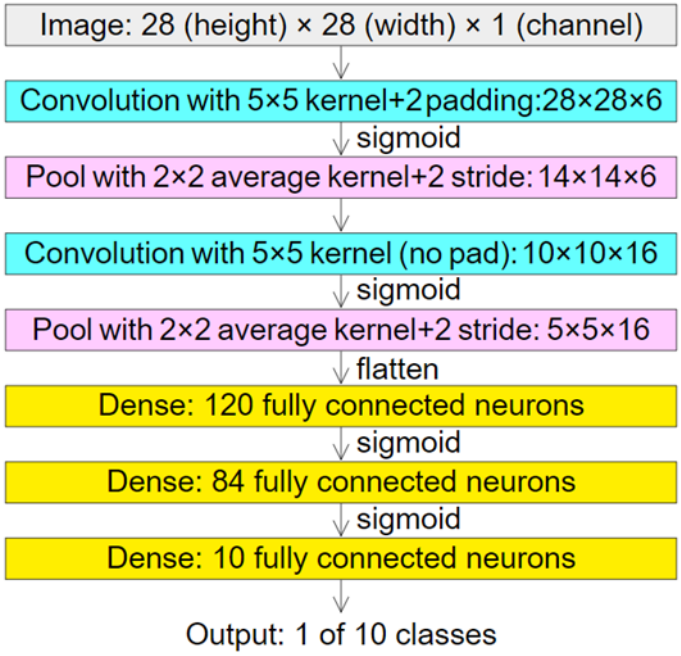

Figure C.1: LeNET architecture [source : wikipedia]

- Expected image size of 224 x 224 with 3 channels.
- Then, convolution of size 11 x 11 with stride 4 is applied on image before 3 x 3 max pooling.
- After this, convolution of size 5 x5 with stride 1 and padding 2 is applied on output of previous layer.
- Then 3 kernel of 3 x 3 which are having stride of 1 and padding of 1 is applied. That are followed by max pooling layer of 3 x 3 with stride of 2.
- Lastly, 3 fully connected dense layer is used.
- In AlexNet ReLU is used as non-linearity.
- Dropout is first used it this architecture. For providing better generalization.

advantage

- More learn-able parameters means that it has higher capacity so in this way it reduce problem of under fitting in this architecture.
- First to use dropout to reduce chance of being overfit.
- First architecture to incorporate ReLU as non-linearity which is far better than its counterpart tanh and sigmoid.

disadvantage

- It took comparatively more time since it has more number of parameter but during that time training will be done on GPU thus training became more faster than before and hence now it is possible to create more deeper architecture

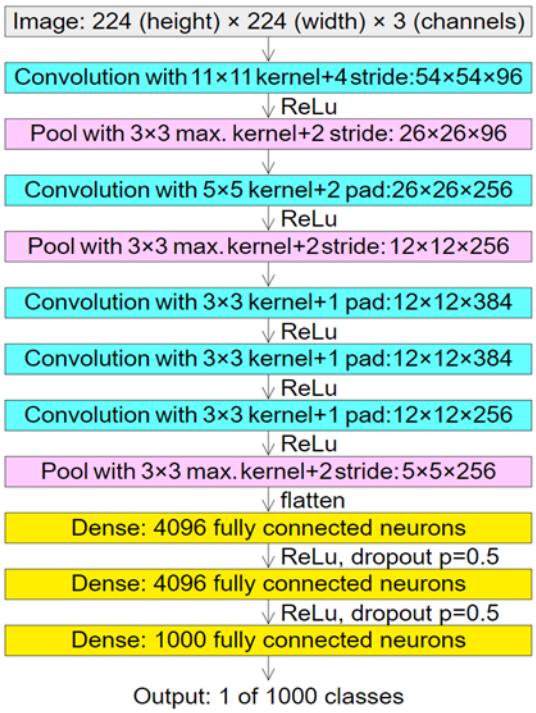

Figure C.2: Alexnet architecture [source : wikipedia]

VGG(2014).

- 13 Conv + 3 Connected Layers with ReLU as activation.
- Max pooling layer of size 2 x 2 is used in architecture.
- Lastly 3 fully connected dense layer is used.
- ReLU was used as non-linearity.
- It consists of 138M parameters

advantage

- Deeper architecture then previous one

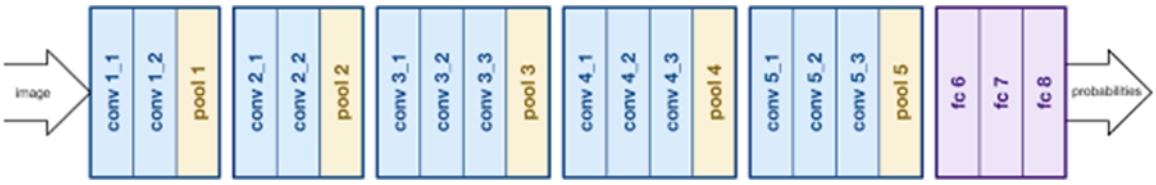

Figure C.3: VGG architecture [source : wikipedia]

- Deeper network means that it has more capacity thus it can get high accuracy on target datasets

Inception-v1(googLeNet)

- Salient features in image can have extremely large variation in size. Suppose that there are two picture of man, one is captured with mobile phone while other is captured with surveillance camera. Both image have same persons, however size of salient features were different.
- Because of this huge variation in size of salient features choosing right kernel size is difficult. Small sized kernel is preferred for information that is distributed more locally while a larger kernel size is preferred for information that is distributed more locally
- So, to solve this problem multiple sizes filters on same level is used. The network essentially would get a bit wider then deeper As shown in Figure C.4

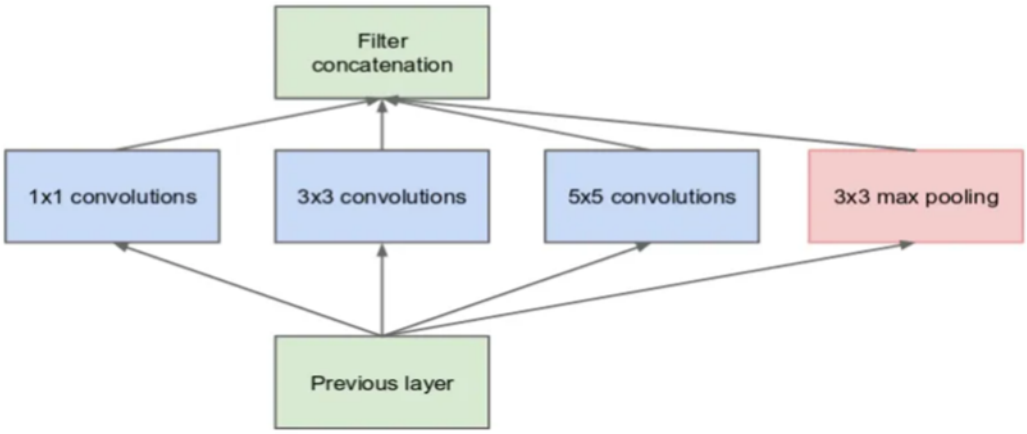

Figure C.4: Inception-v1 naive block [source : medium]

- Applying more convolution operation is not cheaper, it would take significant time so to make it cheaper, the authors limit the number of input channels by adding an extra 1 x 1 convolution before 3 x 3 and 5 x 5 convolutions as shown in Figure-C.5. However it may seems counter intuitive, 1 x 1 convolution with 5 x 5 is more efficient than only single 5 x 5, because 1 x 1 greatly channel size of input.

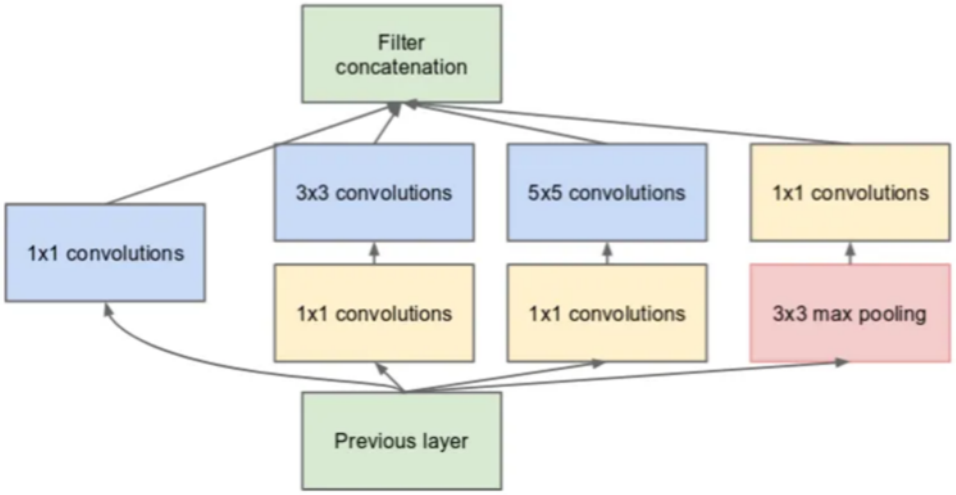

Figure C.5: Improvement on naive block [source : medium]

- Using the dimension reduced inception module, a neural network architecture was built. This was popularly known as GoogLeNet.
- 9 such inception modules stacked linearly. As it was very deep network, it was subject to the vanishing gradient problem. To prevent the middle part of network dying out, the authors of GoogLeNet introduced two auxiliary classifiers. At the end weighted auxiliary loss over same label is calculated. Here weight for auxiliary loss was about 0.3.

$$total_{loss} = real_{loss} + 0.3 \times aux_{loss_1} + 0.3 \times aux_{loss_2} \tag{C.1}$$

Inception-v2

- Inception v2 and Inception v3 were presented in the same paper. The authors proposed a number of upgrades which increased the accuracy and reduce the computational complexity.
- The premise of inception v2 was to reduce representational bottleneck. Neural network perform better when convolutions didn't alter the dimension of the inputs too much. Reducing the dimensions to great extent will cause loss of information, known as a "representational bottleneck".
- And second premise was, using smart factorization methods, convolutions can be made more efficient in terms of computational complexity.
- To solve this issue, 5 x 5 convolution in inception v1 is replaced with two 3 x 3 convolutions to improve computation speed. Moreover, authors factorize filter size n x n to combination of 1 x n and n x 1 convolutions.

- All the outputs and feature maps from convolution filters are combined into one object to create a single output of the inception module by filter concat.
- So inception v2 has following inception module.

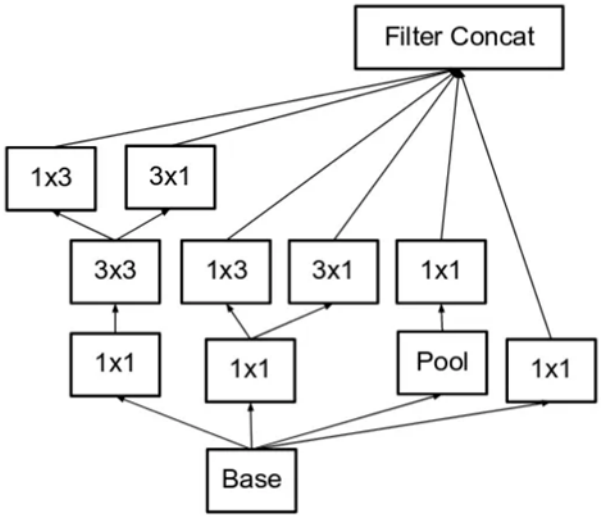

Figure C.6: inception-v2 block [source : medium]

Inception-v3

- Authors noted that auxiliary classifier did not contributed much until near the end of the training process, when accuracy were near saturation. For that they use batch normalization in auxiliary classifier branch.
- Inception Net v3 incorporated all of the above upgrades stated for inception v2, and in addition used the following.
  - RMSProp optimizer
  - Factorized 7 x 7 convolutions.
  - Batch Normalization in auxiliary classifier

Inception-Resnet

- Inspired by the performance of the ResNet, a hybrid inception module was proposed. There are two sub-versions of Inception-ResNet, namely v1 and v2.
- Both the versions have different stems.
- Adding residual connection means that output of convolution operation added with input.

- To provide stability, authors scaled the residual activations by value around 0.1 to 0.3. With means that output of inception module is getting multiplied with above number before it was getting added with input.

Xception(2017)

- Xception by google, stands for extreme version of inception.
- Depth wise separable convolution is used which was proved to better than inception-v3 for both ImageNet ILSVRC and JFT datasets.
- Depth wise convolution is used rather than conventional one.
- **Depth-wise convolution**
  - o Depth-wise convolution is the channel-wise n x n spatial convolution. Which means that suppose image have 3 channels then in that case it had 3 n x n spatial convolution. Time it will take is higher than what usually used convolutions
  - And To change dimension of input channel 1 x 1 pointwise convolution is used

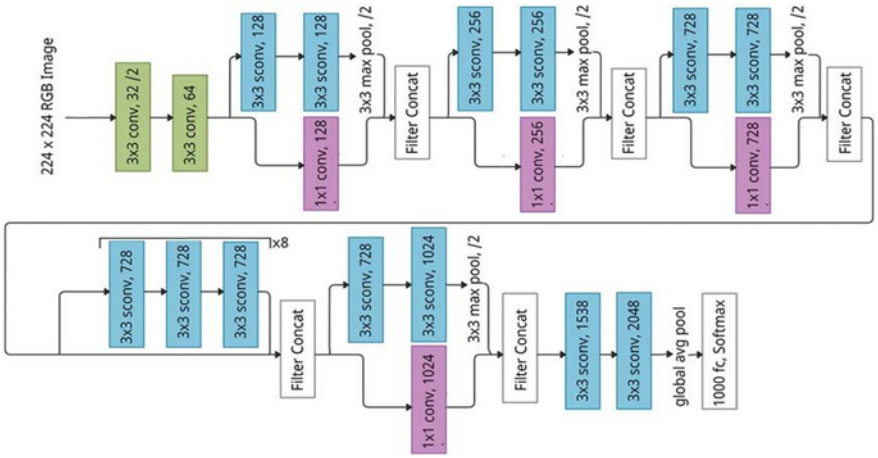

Figure C.7: Xception architecture [source : wikipedia]

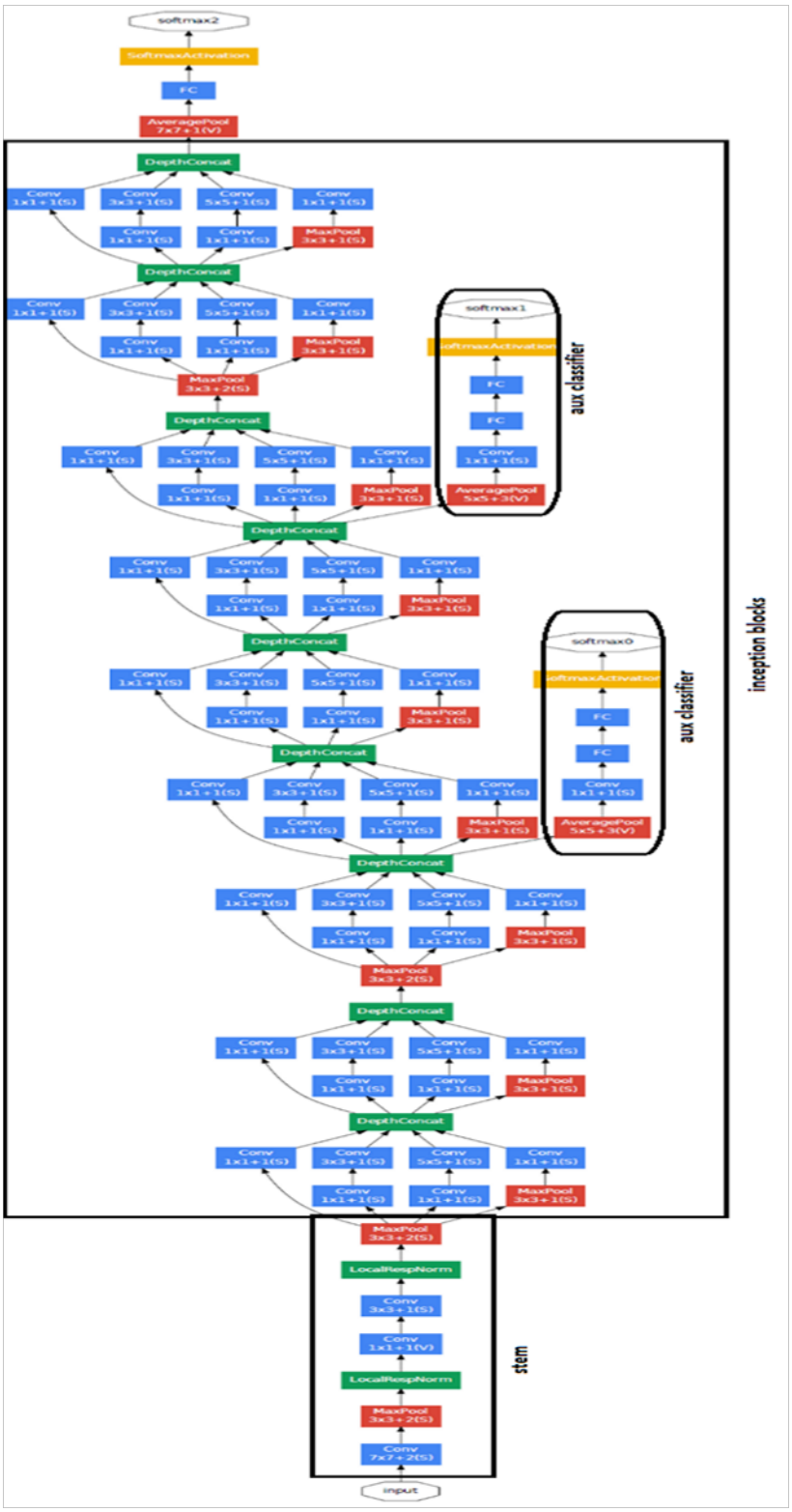

Figure C.8: Inception architecture [source : wikipedia]

Appendix D

Hyperparameters training Figures

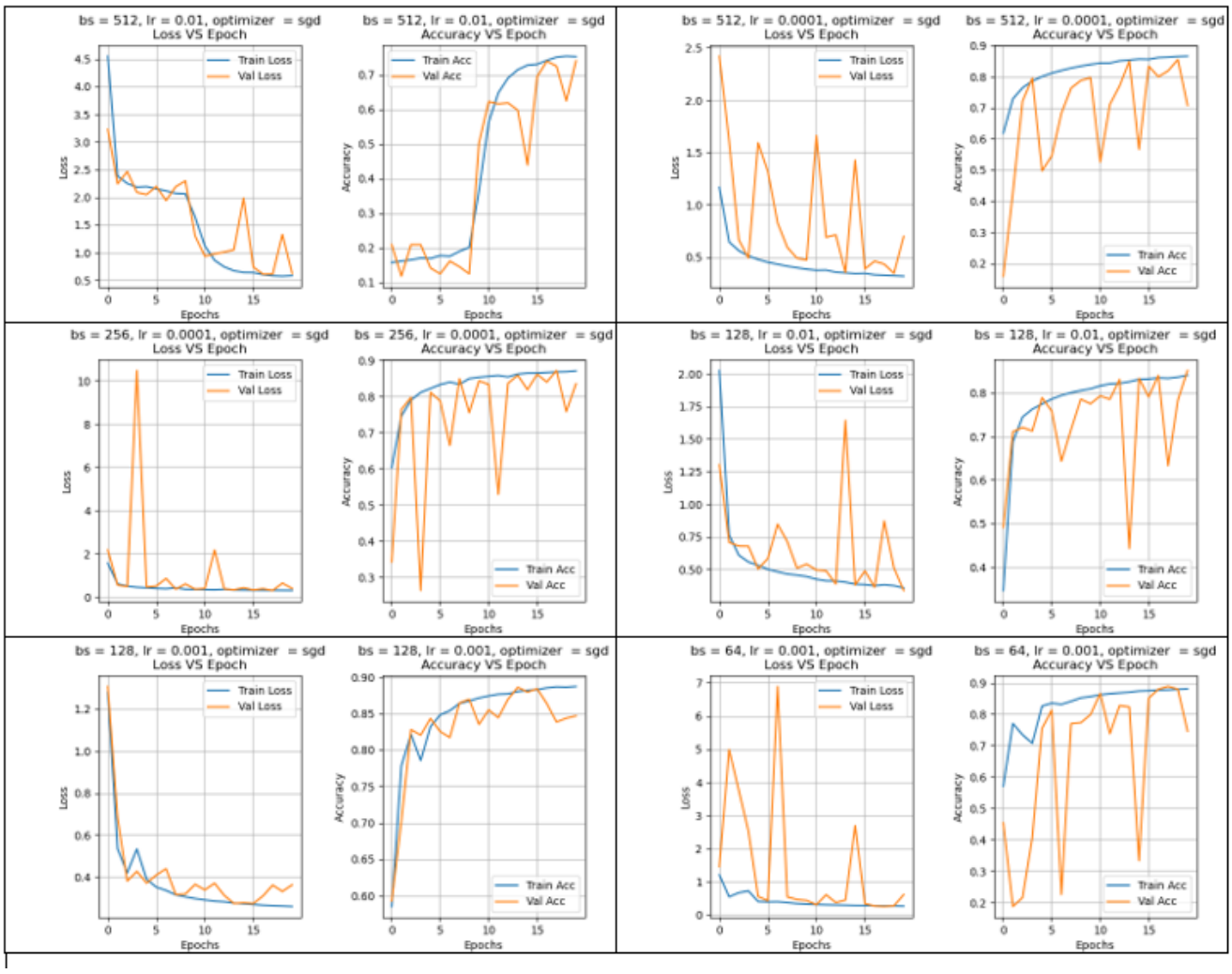

Figure D.1: convLSTM hyperparameter result on SGD

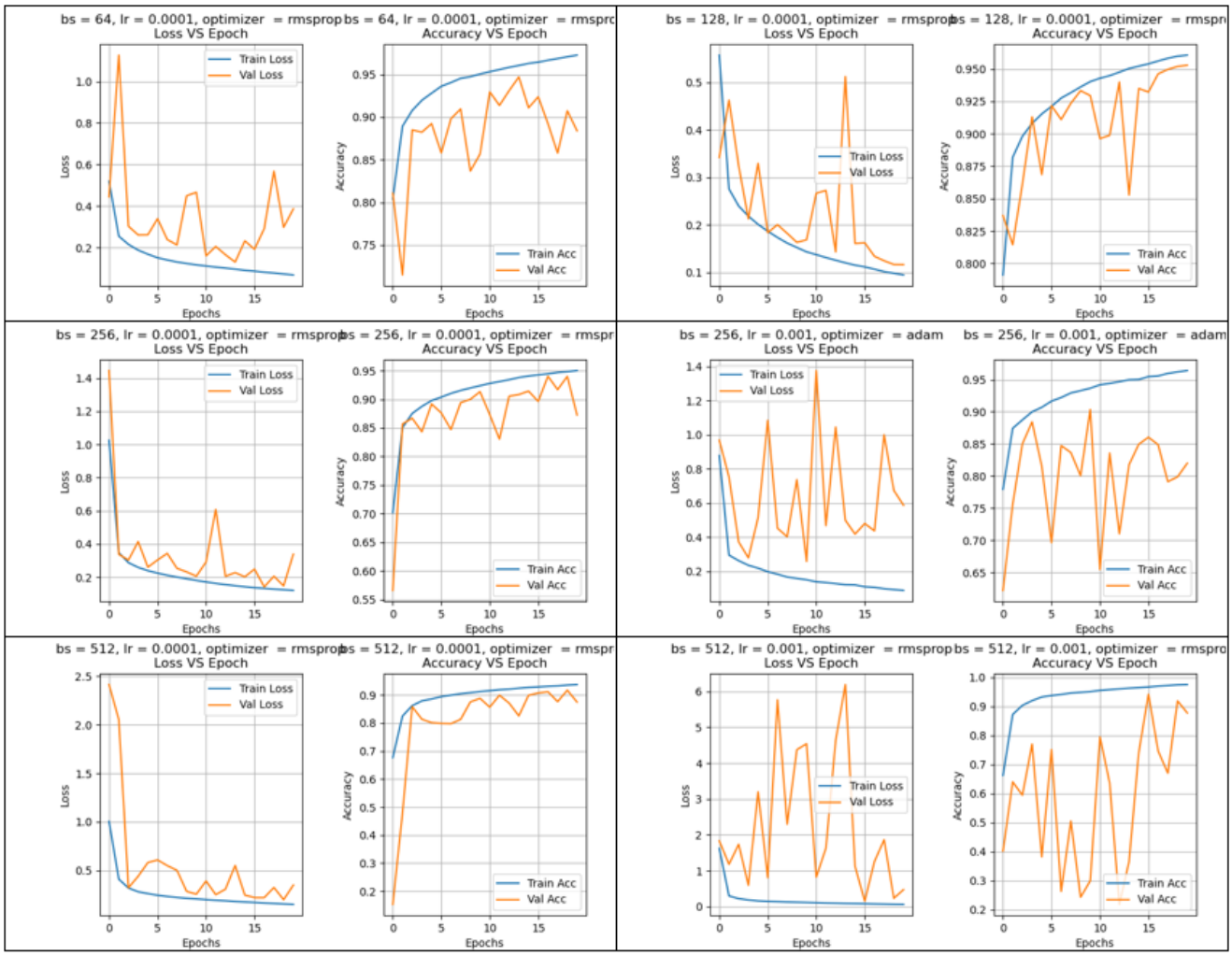

Figure D.2: convLSTM hyperparameter result on RMSprop

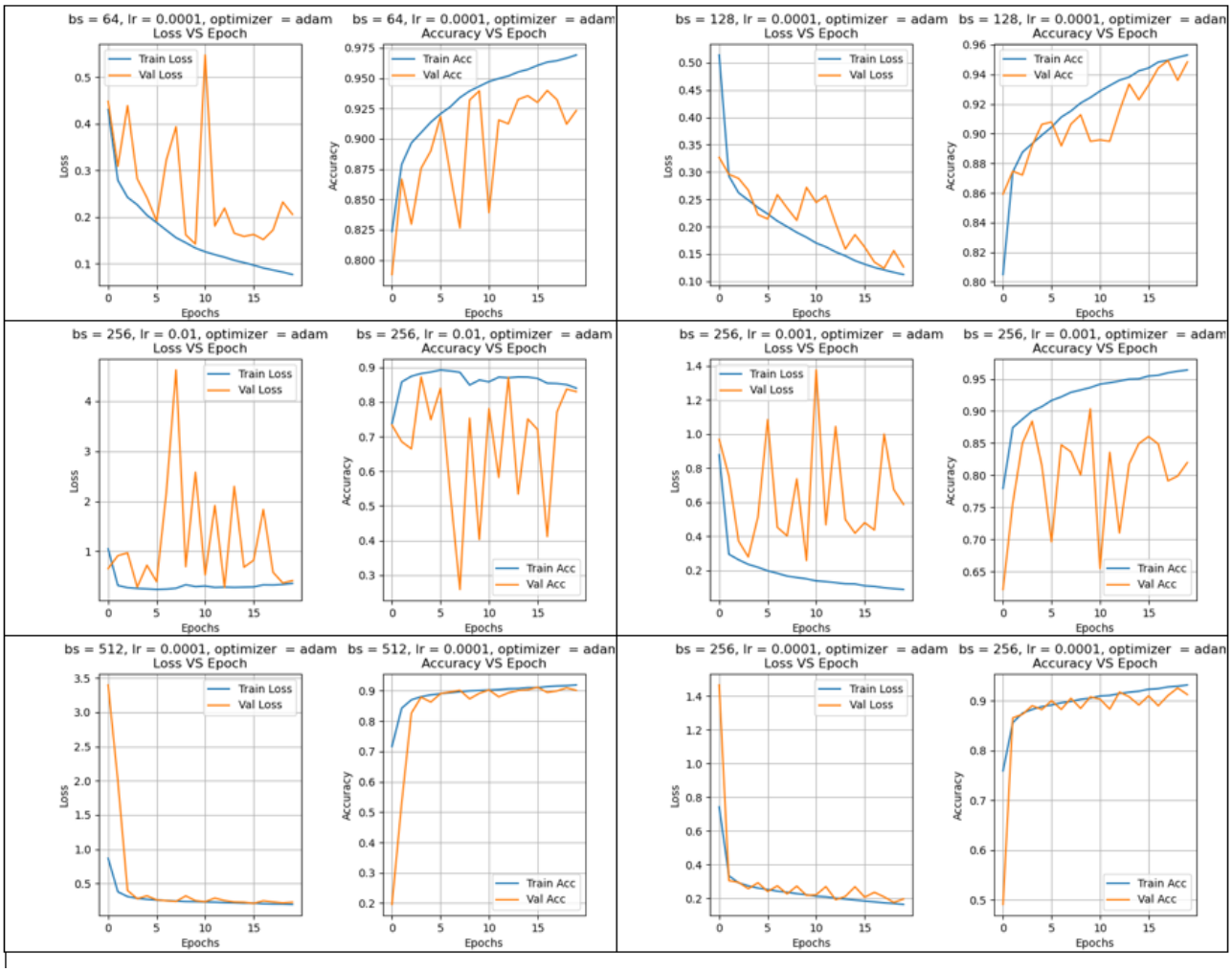

Figure D.3: convLSTM hyperparameter result on Adam

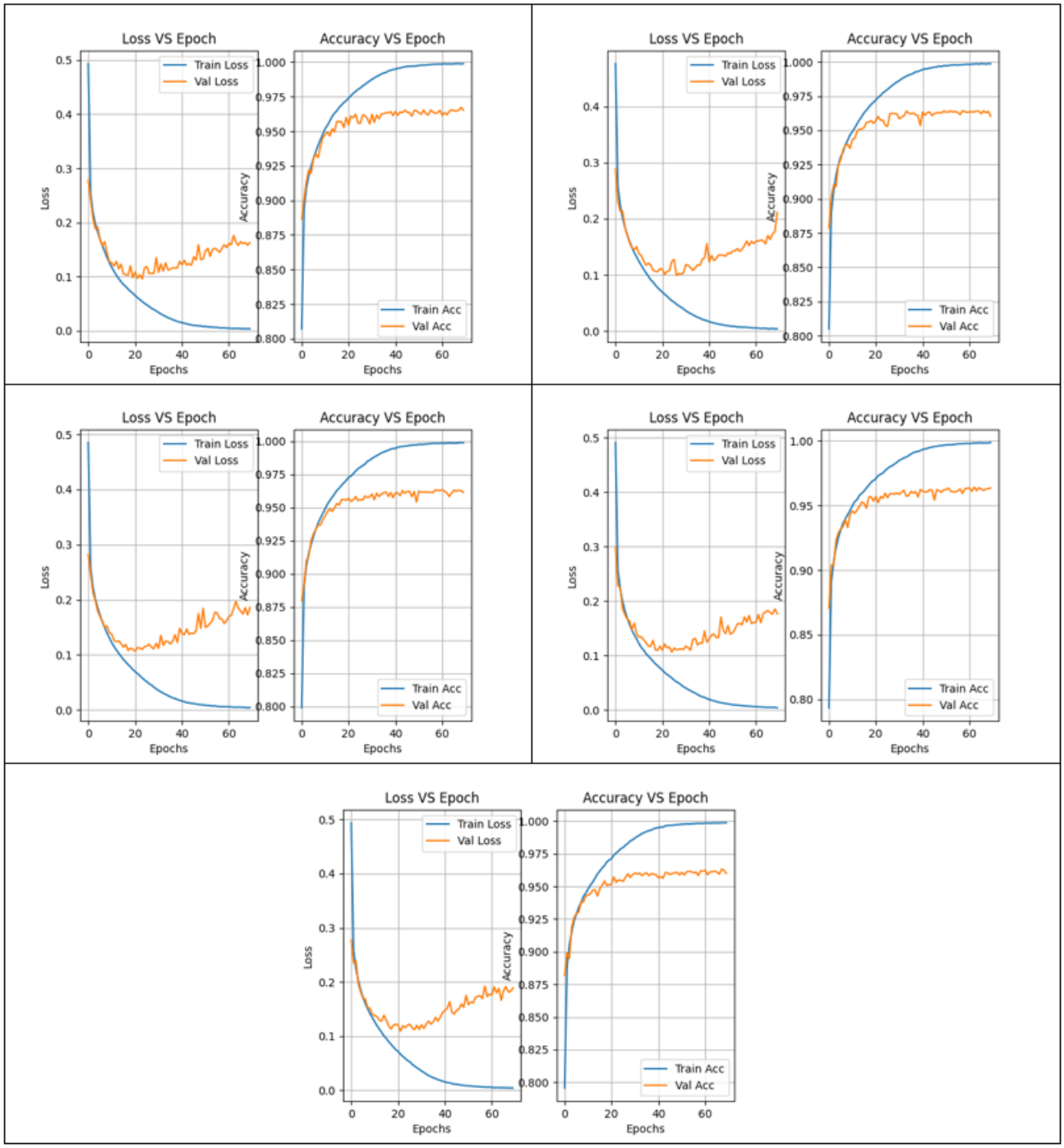

Figure D.4: K-fold result on convLSTM

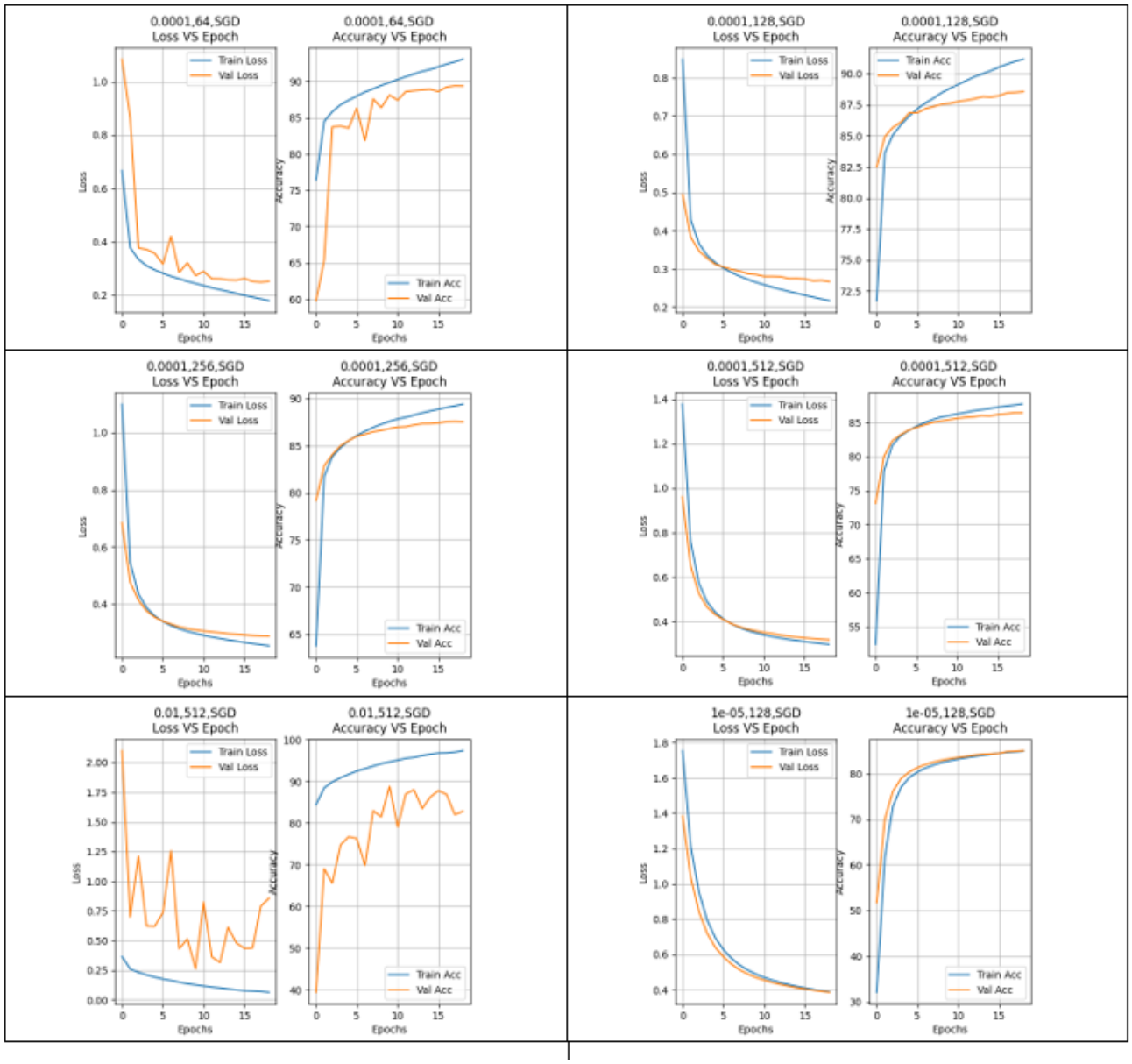

Figure D.5: Resnet hyperparameter result on SGD

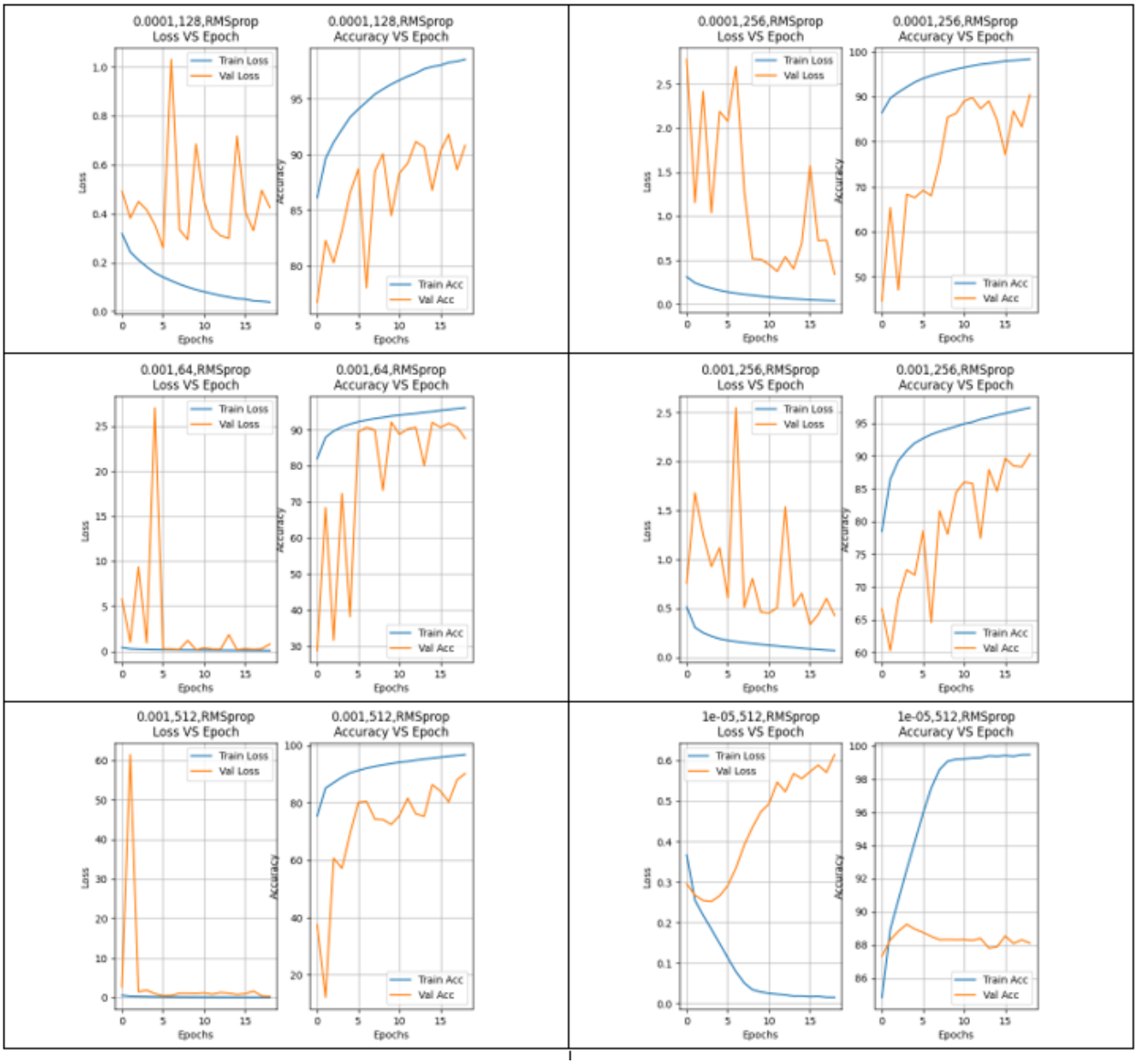

Figure D.6: Resnet hyperparameter result on RMSprop

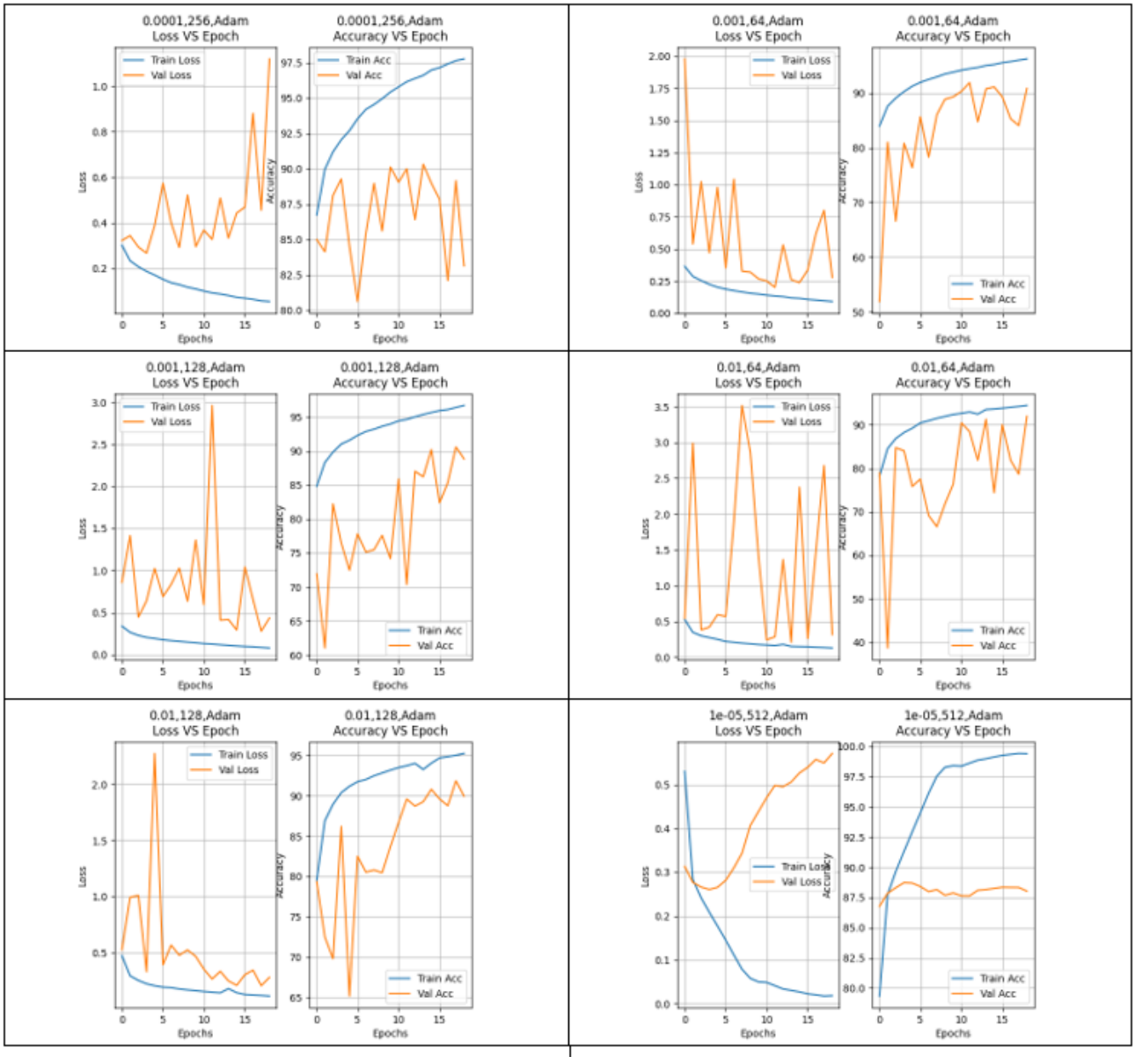

Figure D.7: Resnet hyperparameter result on Adam

Appendix E

Pre-processing Extra Figures

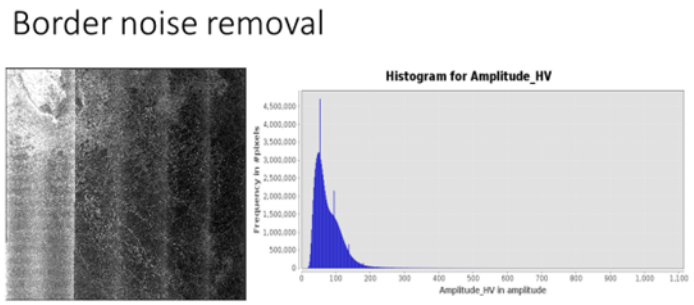

Figure E.1: Border noise removal on HV channel

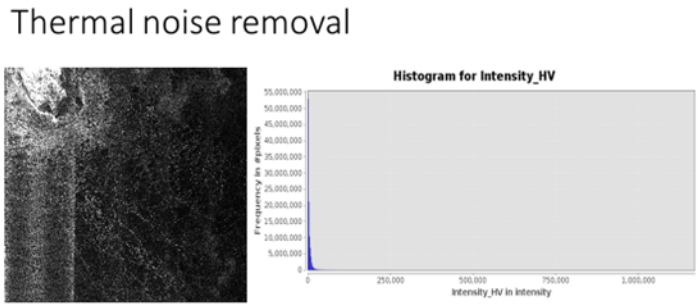

Figure E.2: Thermal noise removal on HV channel

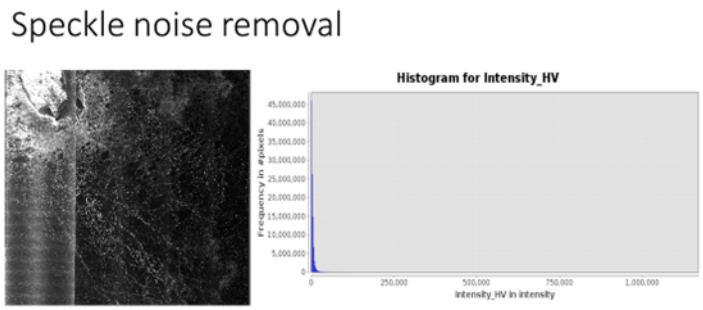

Figure E.3: Speckle noise removal on HV channel

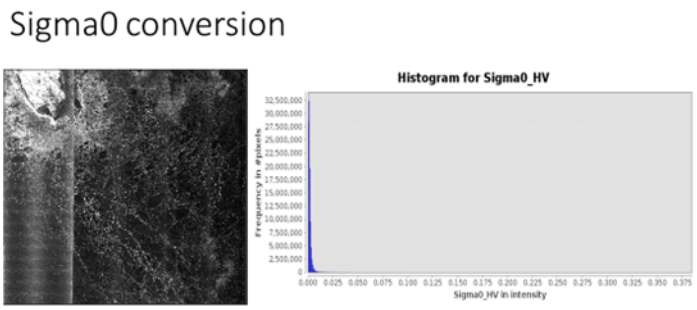

Figure E.4: applying sigma0 on HV channel

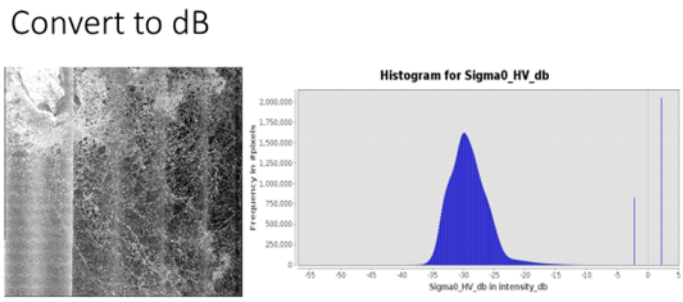

Figure E.5: converting to dB

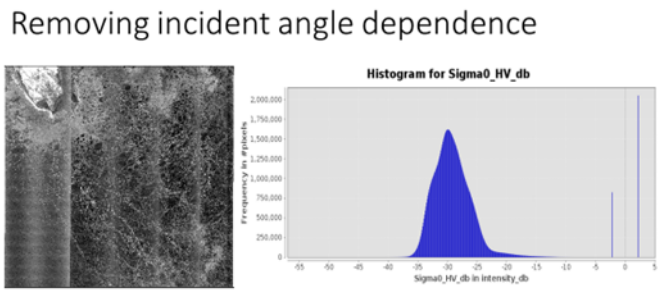

Figure E.6: incident angle dependence correction on HV channel

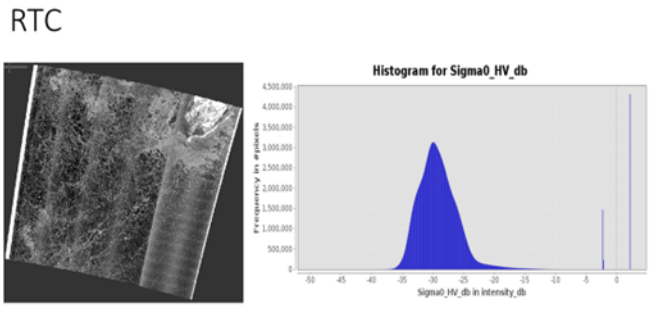

Figure E.7: Range-doppler-terrain correction on HV channel

Appendix F

visualising weights of filters

For visualising weights of filters, we applied filter on SAR image and then analyze output image. Figure-F.1 contain image on which we applied filter.

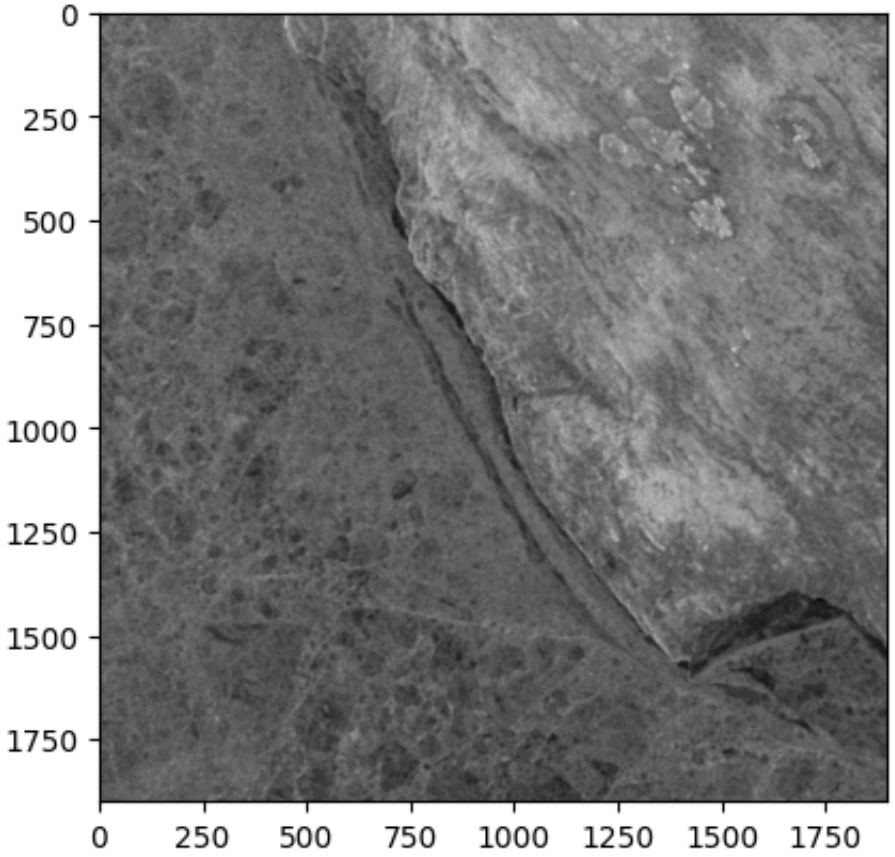

Figure F.1: Original image on which filter is applied

Figure-F.2 contains filtered images. We can clearly see that each filter has significance, on extracting information, as some filter finding edges from the images while some filter changing

intensity. Here we have applied First layer filters of trained Resnet-34 on Figure-F.1 image.

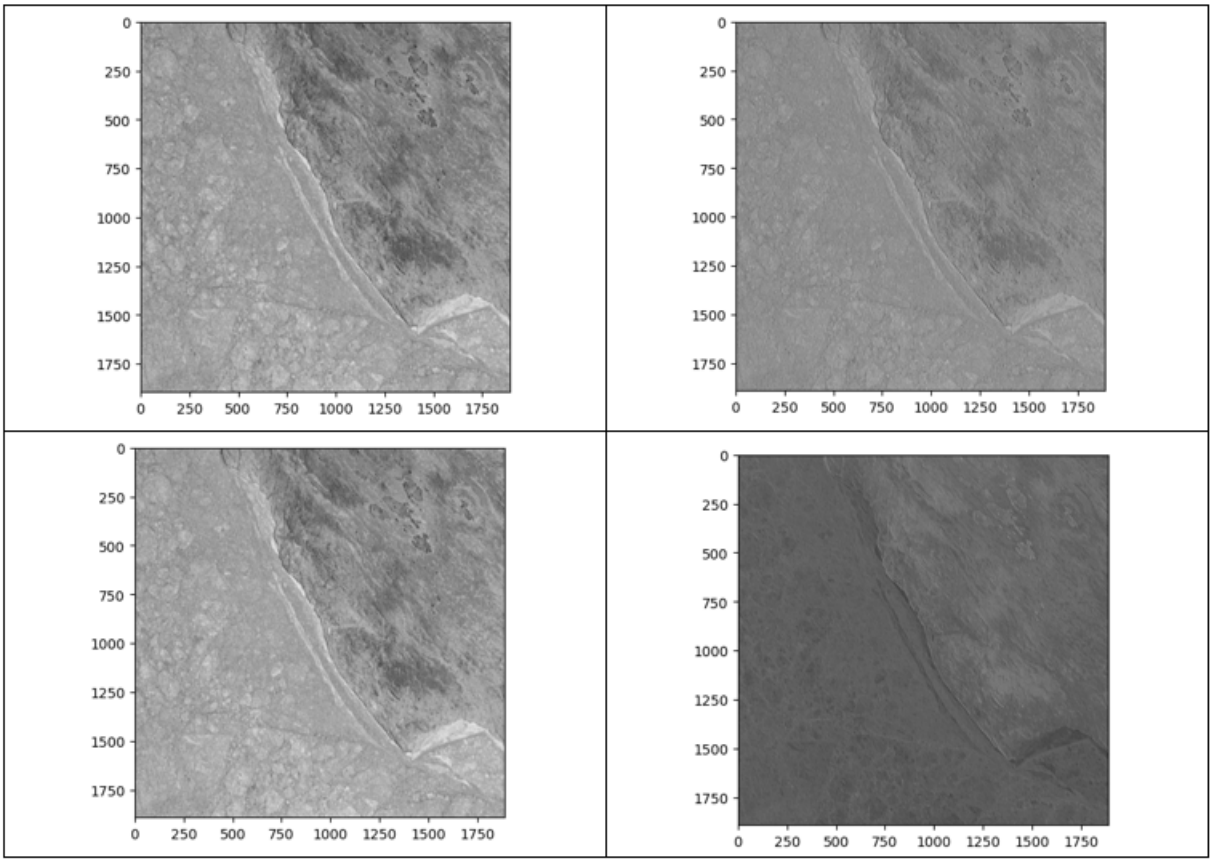

Figure F.2: Filtered Images

For visualizing the filter(kernel), we plot first convolution layer's weights of trained Resnet34. Figure-F.3 contains plot of convolution layer's weights. Image that are present in top left corner of Figure-F.2 is output after applying first kernel($1st$ row, $3rd$ column in Figure-F.3). Image at bottom left is output after applying eleventh kernel($2nd$ row, $3rd$ column in Figure-F.3). Furthermore, Image at top right and bottom right is output after applying kernel 6($1st$ row, $6th$ column in Figure-F.3) and kernel 12($2nd$ row, $4th$ column in Figure-F.3) respectively.

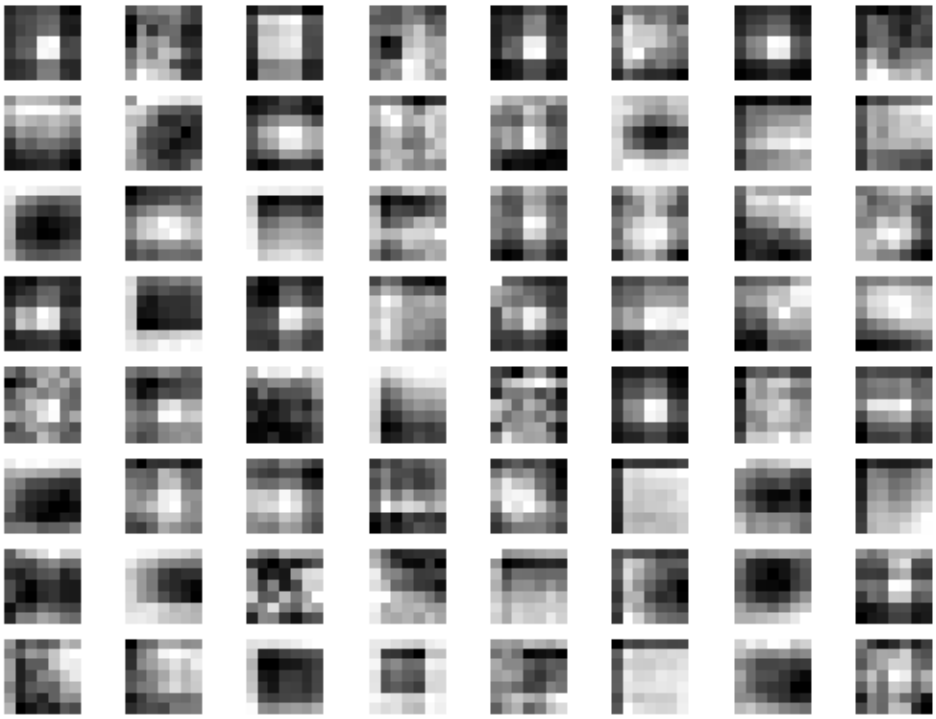

Figure F.3: Trained Resnet layer-1 convolution weighs