Title

Novel Methods for Enhancing Skin Cancer Classification

Professor:

Dr. Mohammed Kayed

Assistant professor:

Dr. Abdelrahim Koura

Team members:

Mohamed Aboukhair (TL)

Amr Hesham Hosni

Mostafa Kamal

Mohamed Salah

AKNOWLEDGEMENTS

Instructors

we would like to express our deepest appreciation to our professors Dr. Mohammed Kayed and assistant professor Dr.Abdelrahim Koura for giving us valuable advice, invaluable contribution, Share of their time with us, their constructive criticism, their suggestions, their unwavering support, their guidance and share their experience as well as their patience that cannot be underestimated and theirs profound belief in our abilities during our project

Helpers

We'd like to acknowledge and appreciate the effort of Osama Hefny too for his support, his helpful advice and his profound belief in our work

Friends

We would like to thank our true friends for giving us encouraging words to keep up trying to reach results when we get stuck in problems

Contents

Abstract

Background:

classifiers based on convolutional neural networks (CNNs) were shown to classify images of skin cancer on par with dermatologists, could enable lifesaving and fast diagnoses, even outside the hospital via the installation of apps on mobile devices, we decide to retake ISIC challenge 2018 to improve our knowledge and to do brainstorm to find solutions to the problem that we would meet by new experimental methods in that field because it is lifesaving, so we start to research about best methods for skin cancer classification, we found papers that classified skin lesions using CNNs. In principle, classification methods can be differentiated according to three principles.

Approaches that use a CNN already trained by means of another large dataset and then optimize its parameters to the classification of skin lesions are the most common ones used and they display the best performance with the currently available limited dataset, CNN's display a high performance as skin lesion classifiers.

Unfortunately, it is difficult to compare different classification methods because some approaches use nonpublic datasets for training and/or testing, thereby making reproducibility difficult, that is why we only used challenge dataset without any addition of any other dataset even if it is public because we doubt its existence later, we could figure out 2 functions one for segmentation and another that can improve both ensembles in stacking models predations or single model results.

Objective:

This study aims to retake ISIC challenge 2018 with its dataset with CNNs using a new experimental method that solves some of the current problems that we have met and try to improve it and compare it to common methods. We limit our research to the ISIC challenge 2018 dataset. In particular, methods that apply a CNN only for segmentation or for the classification we consider both. Furthermore, this study discusses methods applied on skin cancer dataset that have big imbalance which is very difficult and which have challenges that we have to face to improve results while test methods that we propose as solutions to improve common methods results.

Results:

we could achieve 83.4% by our experimental method called ‘Class weight transformation’ compared to first place person that achieves 84.5% by only using public data same as us but he used more computation power in training single model as well as we could improve the method for image segmentation called ‘limited crop certain’ without losing data compared by normal segmentation bitwise with mask or crop only cancer from the image that improved results of segmentation by more over than 3% to 5% in each model we trained.

Conclusions:

by brainstorm each problem that we faced we could come out with 2 methods that improve segmentation results for training stage and delete noise that might happen for train stage after using segmentation as pre-processing, we could make a method that may improve results of a single model and as well as ensemble results from the prediction of models that classified under stacking model prediction methods like vote, average and weighted average in ensemble methods.

Chapter 1: Introduction

Overview:

Skin cancers result in 80,000 deaths a year as of 2010, 49,000 of which are due to melanoma and 31,000 of which are due to non-melanoma skin cancers [1], This is up from 51,000 in 1990 [1], In the US in 2008, 59,695 people were diagnosed with melanoma, and 8,623 people died from it. [2] In Australia more than 12,500 new cases of melanoma are reported each year, out of which more than 1,500 die from the disease. Australia has the highest per capita incidence of melanoma in the world [3], Because of that the more results that method gives in skin cancer classification the more people's lives saved from this dangerous disease.

We used CNN [41] as our main extractor to replace making computer vision algorithms for each feature that we need to extract from each cancer, then we made the segmentation model test it with our methods.

Objective:

The main objective of this study is to develop a method that increases classification normal method results in skin cancer detection in order to achieve that we need to compare results of normal methods with our experimental methods within the limited dataset and see improvement of each method.

Skin Cancer Classification:

State-of-the-art classifiers based on convolutional neural networks (CNNs) were shown to classify images of skin cancer on par with dermatologists and could enable lifesaving and fast diagnoses, even outside the hospital via the installation of apps on mobile devices.

CNN's display a high performance as state-of-the-art skin lesion classifier. Unfortunately, it is difficult to compare different classification methods because some approaches use nonpublic datasets for training and/or testing, thereby making reproducibility difficult. Future publications should use publicly available benchmarks and fully disclose methods used for training to allow the comparability.

Motivation to do this work:

This problem still open for the search to find new methodologies

Try and test our new methods with skin cancer: -

1- Limited crop certain: is one of our experimental method that was the solution for crop certain noise by a limit box that contains cancer to be at least at the size of our resize method size (224,224) plus the size of the error that if we want to add it we will sum it with resizing in our case was 10 pixels for each min and max of width which make total is 224+20=244 then it would be no problem with resizing bigger image as far we would explain below more details in pre-processing chapter

2- Class weight transformation: a method for ensemble models instead of giving weight to each model we give weight to each class in every model, it has more details in the training chapter

Compare methods:

We choose some methods to compare ours by those common methods, most of them compare methods were good enough to compare and gain information that we need but we notice some strange behavior that only can be explained in training stage because what lead to this strange behavior was training stage for each model.

Methods to compare our experimental methods:

1-) We would compare the first method of segmentation limited crop certain with mask bitwise and explain why did we make it limited not crop certain as it is without change and we would explain how crop certain could add noise in the training stage:
   i. mask bitwise with the real image to show only cancer
   ii. crop certain noise and disadvantage

2-) we would compare second method Class weight transformation with stacking ensemble methods and will explain how this method could improve test score for a single model:
   i. voting stacking ensemble
   ii. average stacking ensemble
   iii. weighted average by using a genetic algorithm as optimizer that made strange behavior

Limitation & challenges:

1-) we face a big imbalance in classes' distribution
2-) there are different sizes of cancer of the same class so it is hard to extract similar features to classify it
3-) not all the images have the same light effect
4-) cancers from different classes have similar patterns so it makes it hard to differentiate
5-) each class have many patterns and it will be shown in skin cancer section
6-) challenges facing in this field are the inability to obtain private datasets easily and publicity of these datasets
7-) limited resources of colab

The proposed models: -

We train 7 models for each version of pre-processing versions

Models: (densenet201, densenet169, resnet152, resnet101, vgg16, vgg19, senet154)

We choose common models and different versions of them to see different effects of each version of pre-processing on each model.

Tools and environment used

Python in google colab for online GPU as environment, fastai library (main tool), PyTorch library, TensorFlow library.

Chapter 2 skin cancer and dataset

Introduction

Reason:

As we said in the limitation, we decided we are going to show patterns and structures and features of skin cancer to prove our point of view

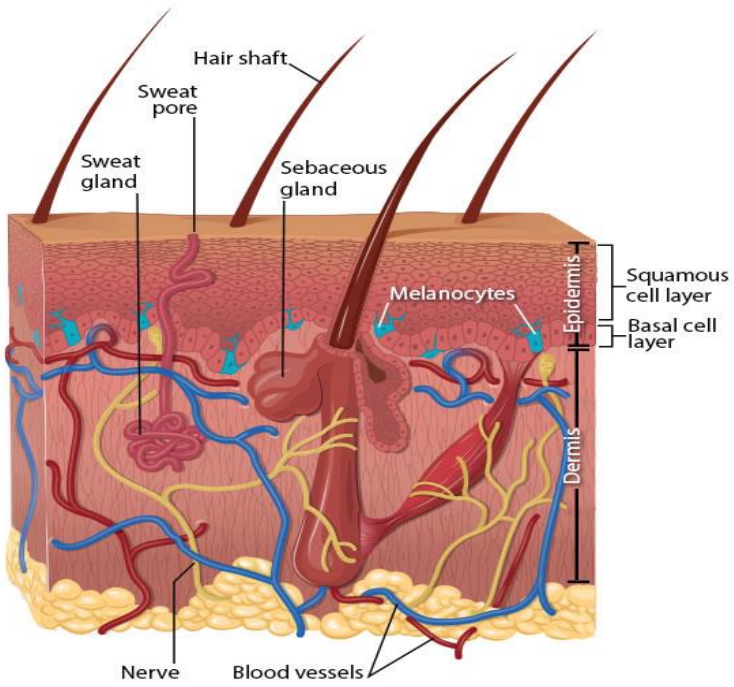
Skin cancer

Overview:

Cancer is a disease in which abnormal cells can invade or spread to other parts of the body [9]. When cancer starts in the skin, it is called skin cancer.

Skin cancer is the most common cancer in the United States. Some people are at higher risk of skin cancer than others, but anyone can get it. The most preventable cause of skin cancer is overexposure to ultraviolet (UV) light, either from the sun because more than 90% of cases are caused by exposure to ultraviolet radiation from the Sun [6], this exposure increases the risk of all three main types of skin cancer. [6] Exposure has increased, partly due to a thinner ozone layer. [7][8] Tanning beds are another common source of ultraviolet radiation. [6]

The two most common types of skin cancer—basal cell and squamous cell carcinomas [5]—are highly curable, but can be disfiguring and costly to treat.



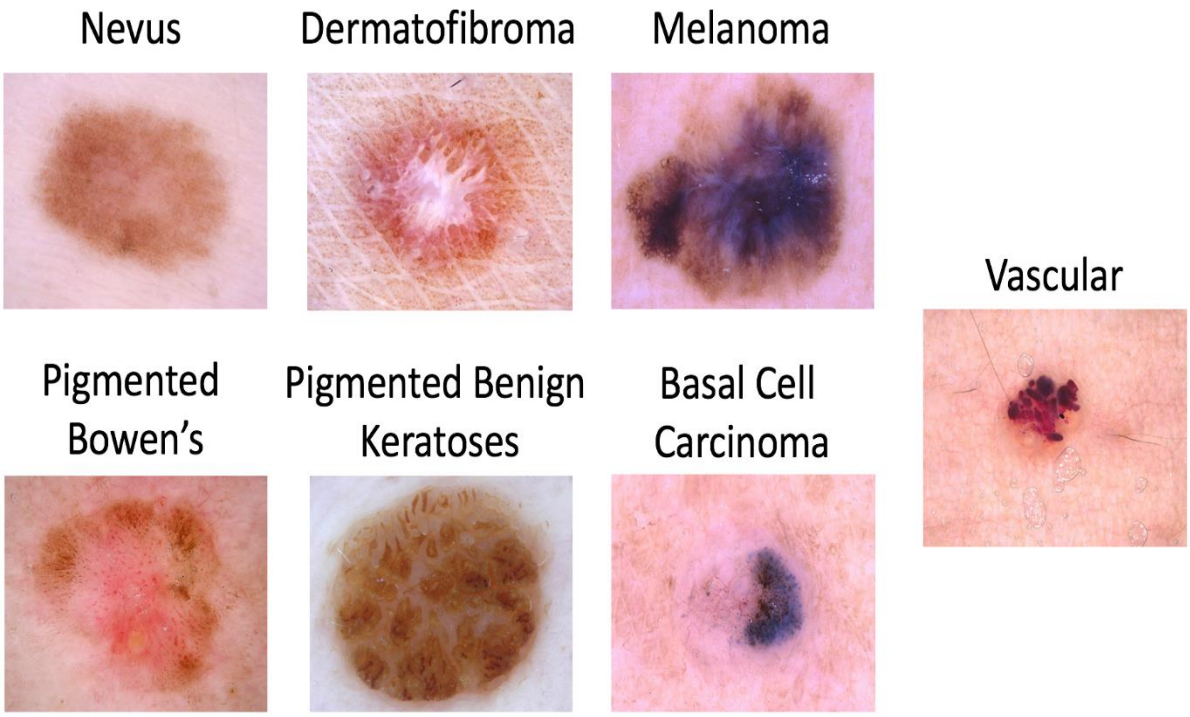
Skin layer (figure 1)

Melanoma, the third most common skin cancer, is more dangerous and causes the most deaths, Melanomas are the most aggressive. Signs include a mole that has changed in size, shape, color, has irregular edges, has more than one color, is itchy, or bleeds. [10]

Anyone can get skin cancer, but people with certain characteristics are at greater risk—People with lighter skin are at higher risk [5] [11] as are those with a poor immune function such as from medications or HIV/AIDS.[7][12] The diagnosis is by biopsy[10]. A family history of skin cancer. A personal history of skin cancer.

Regardless of whether you have any of the risk factors listed above, reducing your exposure to ultraviolet (UV) rays can help keep your skin healthy and lower your chances of getting skin cancer in the future. Most people get at least some UV exposure from the sun when they spend time outdoors. Making sun protection an everyday habit will help you to enjoy the outdoors safely, avoid getting a sunburn and lower your skin cancer risk.

Types of cancers in our dataset



Type of cancers (figure 2)

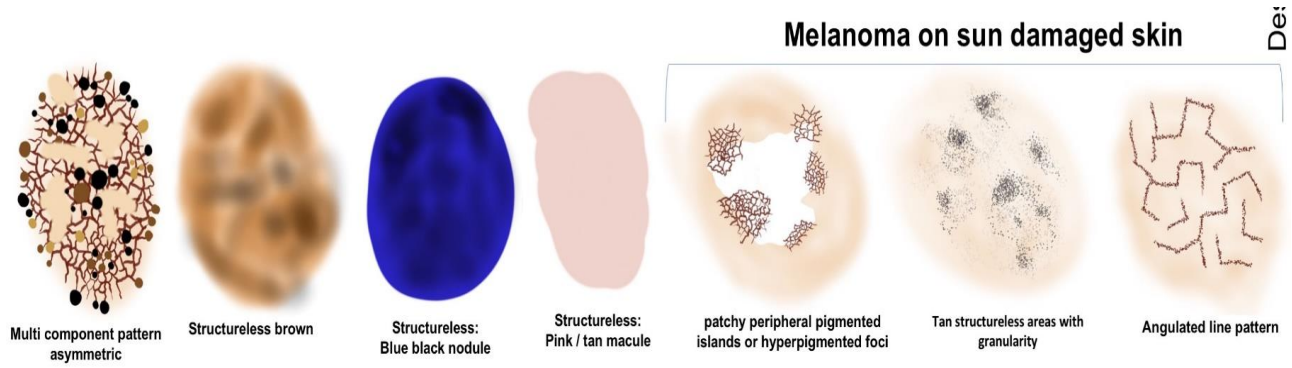
MEL: "Melanoma":

Malignant melanoma has many faces and depending on the type of melanoma different dermoscopy criteria can be seen

Melanomas will usually manifest a disorganized distribution of structures and colors making their identification quite easy. These lesions will reveal at least one, but usually more than one, of the melanoma-specific structures listed below. On rare occasions, melanomas will present with an asymmetric and organized pattern but these tumors will almost always reveal one of the following features: starburst pattern, negative network, blue-black or gray color, shiny white structures, vessels or ulceration. A few melanoma patterns deserve special mention. [13]

Melanoma patterns:

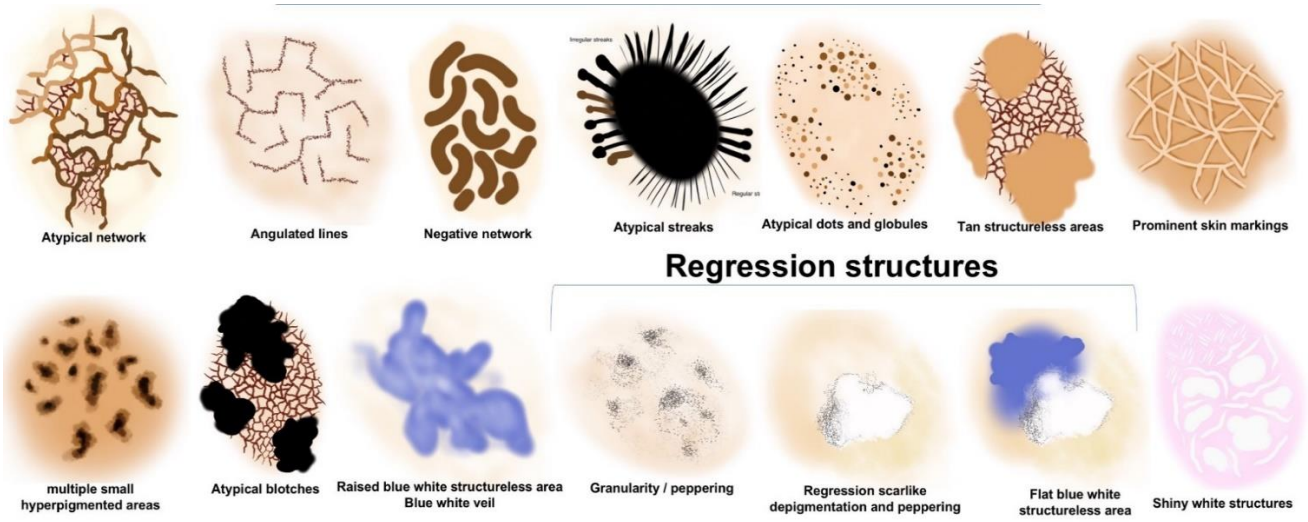
There are seven patterns in melanoma [multi component pattern asymmetric, structureless brown, structureless: blue-black nodule, structureless: pink/tan macule, patchy peripheral pigmented islands, tan structureless areas with granularity, angulated line pattern] as shown in figure 3.



Melanoma patterns (figure 3)

Structures:

There are fourteen structure in melanoma [a typical network, angulated lines, negative network, a typical streak, a typical dots and globules, tan structureless areas, prominent skin markings, multiple small hyperpigmented areas, typical blotches, raised blue-white structureless area blue-white veil, granularity/peppering, regression scar-like depigmentation and peppering, flat blue-white structureless area, shiny white structures] as shown in figure 4.



Melanoma Structures (figure 4)

NV: "Melanocytic nevus"

Melanocytic nevi are commonly classified based on a mix of clinical and histopathological criteria. Clinically nevi have been initially classified as either acquired or congenital, both further subdivided into a junctional, compound, and dermal naevi based on the location of the nests of melanocytes and nevocytes within the skin (epidermis or dermis). It has been shown that one of the keys to recognizing melanoma is knowledge of the many faces of benign lesions. [14]

Melanocytic nevus Patterns

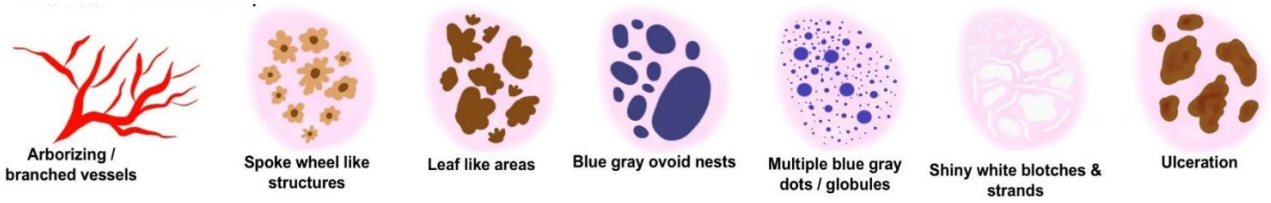
There are five patterns in Melanocytic nevus [Globular (congenital) naevus, Reticular (acquired) naevus, Starburst (Spitz ⁄Reed) nevus, Blue (homogeneous) naevus, Site-related naevi]

BCC: "Basal cell carcinoma"

Basal cell carcinoma (BCC) is the most common type of skin cancer in the world. Although mortality related to BCC is negligible, BCCs can be associated with significant morbidity, especially if left untreated and/or if discovered when they have attained relatively large diameters. Clinically, BCC can present with a variety of morphologies ranging from erythematous patches to ulcerated nodules. There are multiple histopathologic subtypes of BCC including superficial, nodular, morphea form/sclerosing/infiltrative, fibroepithelioma of Pinkus, microcystic adnexal and baso-squamous cell BCC. Each subtype can be clinically pigmented or non-pigmented. It is not uncommon for BCCs to display pigment on dermoscopy with up to 30% of clinically non-pigmented BCCs revealing pigment on dermoscopy. Based on the degree of pigmentation, some BCCs can mimic melanomas or other pigmented skin lesions. Depending on the subtype of BCC and the degree of pigmentation, the clinical differential diagnosis can be quite broad ranging from benign inflammatory lesions to melanoma. Fortunately, the use of dermoscopy has dramatically improved the diagnostic accuracy and diagnostic confidence of clinicians for both pigmented and non-pigmented BCCs (Schematics show the features seen in pigmented and non-pigmented BCC). Also, dermoscopy permits for the diagnosis of clinically tiny BCCs since the dermoscopic criteria for BCC are visible irrespective of the size of the tumor. [15]

Basal cell carcinoma features

There are seven features in Basal cell carcinoma [arborizing / branched vessels, spoke wheel-like structures, leaf-like areas, blue-gray ovoid nests, multiple blue-gray dots / globules, shiny white blotches & strands, ulceration] as shown in figure 5



BCC Dermoscopic features as standalone (figure 5)

AKIEC: "Actinic keratosis / Bowen's disease (intraepithelial carcinoma)"

Actinic (solar) keratosis (AK), Bowen's disease (BD), keratoacanthoma (KA), and squamous cell carcinoma (SCC) comprise the spectrum of premalignant and malignant keratinizing tumors. In contrast to the well-defined dermoscopic criteria of pigmented tumors, the dermoscopic features of these, mostly non-pigmented keratinizing tumors, are less well established. Most of the described dermoscopic patterns are based on case series. The dermoscopic diagnosis of these tumors is mainly based on the assessment of vascular patterns. The architectural arrangement and distribution of the vessels within the lesion and the correlation with the clinical assessment (e.g. texture, firmness) may provide improved specificity. Other associated, but nonspecific features are erythema, scale, erosion, or keratin. Since their diagnosis is mostly based on the ability to visualize blood vessels under dermoscopy, the use of polarized light dermoscopy instruments is preferred as it seems to provide the best method to visualize vascular structures. Also, using a viscous immersion medium, such as ultrasound gel, when applying contact dermoscopy, will allow for better visualization of the vascular structures as it eliminates the effect of pressure-induced compression of blood vessels. [16]

AKIEC structures

There are six structures in AKIEC [Actinic keratosis, Non-pigmented actinic keratosis (facial skin, Non-facial skin), Pigmented actinic keratosis, Bowen's disease, Keratoacanthoma, Squamous cell carcinoma]

BKL: “Benign keratosis (solar lentigo / seborrheic keratosis / lichen planus-like keratosis)”

-Solar lentigo

Solar lentigines are sharply circumscribed, uniformly pigmented macules that are located predominantly on the sun-exposed areas of the skin, such as the dorsum of the hands, the shoulders, and the scalp. Lentigines are a result of hyperplasia of keratinocytes and melanocytes, with increased accumulation of melanin in the keratinocytes. They are induced by ultraviolet light exposure. [20]

Unlike freckles, solar lentigines persist indefinitely. Nearly 90% of Caucasians over the age of 60 years have these lesions. Due to the increased prevalence of lentigines in the elderly, these lesions are sometimes referred to as “lentigo senilis”. However, younger individuals who tend to burn after ultraviolet exposure can also develop lentigines after acute or prolonged ultraviolet light exposure. Clinically, solar lentigines may be oval, round, or irregular in shape and can vary from a few millimeters to a few centimeters in diameter. Most lesions have a uniform light brown color; however, there are instances when they vary from dark brown to black. One variant of solar lentigo, “ink-spot” lentigo, has a jet-black color. [20]

Actinic purpura or other signs of sun damage can frequently be found in the skin surrounding solar lentigines. Solar lentigines are benign lesions that can evolve to a pigmented seborrheic keratosis. Histologically, it is characterized by club-shaped rete ridges with small nub-like extensions. In addition, there is an increased number of melanocytes and increased pigmentation in the basal keratinocytes. Although most solar lentigines are easily recognized on clinical examination, some lesions pose diagnostic challenges because their clinical appearance resembles that of melanoma. Dermoscopy can be helpful in correctly differentiating a solar lentigo from melanoma. [20]

Solar lentigines features

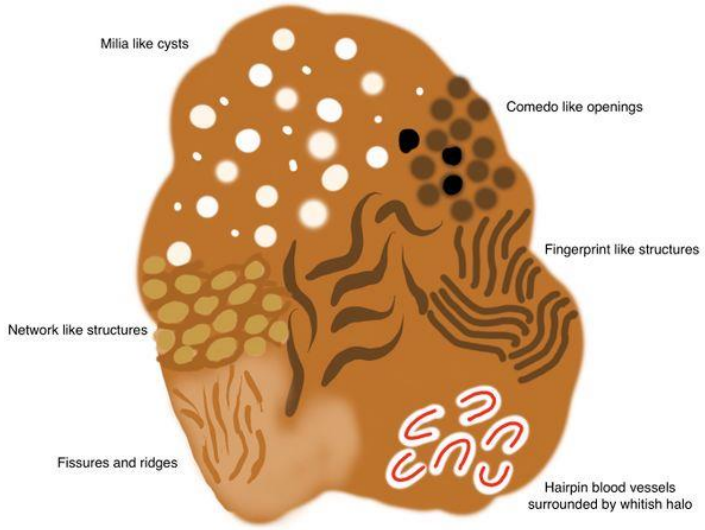
there are seven features in solar lentigines [Moth-eaten border, Homogenous light brown pigmentation, Pigment network, Fingerprint-like areas, Pseudo network, Symmetric brown follicular pigmentation, Ink-sport lentigo]

-Seborrheic keratoses

Seborrheic keratoses are benign epithelial lesions that can appear on any part of the body except for the mucous membranes, palms, and soles. The lesions are quite prevalent in people older than 30 years. The etiology of seborrheic keratoses remains unclear. Ultraviolet light exposure may be responsible for the development of some seborrheic keratoses because they appear to evolve from solar lentigines; however, many develop in areas of the skin naturally protected from ultraviolet light exposure, such as the inframammary (intertriginous) areas. Clinically, early seborrheic keratoses are light- to dark brown oval macules with sharply demarcated borders (solar lentigo). As the lesions progress, they transform into plaques with a waxy or stuck-on appearance. The surfaces of these lesions have a warty and keratotic appearance. Often, the lesions have follicular plugs scattered over their surfaces. The size of the lesions varies from a few millimeters to a few centimeters. Histologically, there are several distinct forms of seborrheic keratoses. In general, the lesions are characterized by papillomatous epidermal hyperplasia of uniform and monotonous keratinocytes and the presence of pseudocysts. The diagnosis of most seborrheic keratoses is straightforward. However, some seborrheic keratoses, especially the deeply pigmented variant, can simulate malignant melanomas. Thin, early lesions have moth-eaten borders and fingerprint-like structures as described for solar lentigines. [20]

Seborrheic keratoses features

There are eight features in seborrheic keratoses [Milia-like cysts, Comedo-like openings, Fissures and ridges, Network-like structures, Cerebriform pattern, Fat-fingers, Sharply demarcated borders, Typical hairpin blood vessels]



Seborrheic keratosis features (figure 6)

-Lichen Planus-like Keratosis

Lichen planus-like keratosis, also known as LPLK and lichenoid keratosis, is one of the common benign neoplasms of the skin. It is believed to be either a seborrheic keratosis or a solar lentigo that is undergoing regression. Supporting evidence has been published beginning with Mehregan's findings of the presence of lentiginous epidermal hyperplasia in lesions interpreted as LPLK. Further supporting evidence can be found by Laur, et al who in 1981 published a detailed clinical-histopathologic correlation in the JAAD. Also, Goldenhersh et al described performing biopsies of lentigines in two instances. The first being a biopsy of a solar lentigo and 5 years later, after the lesion had demonstrated a clinical change into a solitary lichen planus-like keratosis. [20]

Clinical and Histologic Appearance

Lichen planus-like keratosis is a great masquerader with a differential diagnosis including basal cell carcinoma, squamous cell carcinoma, and melanoma. The wide differential diagnosis is due to the extreme variability in characteristic appearance with many pigmentation and morphologic possibilities. The clinical appearance depends on its stage of evolution. [20]

The lesion can appear as a macule or papule that is pink, pinkish brown, pinkish-orange, rust-colored, purplish brown, dusky violaceous, or blue-gray to black. Some lesions are characterized by a velvety appearance, some have a fine-scale, while others have accentuated skin markings. Lesions can be solitary or in some cases multiple. [20]

Early Stage

The histologic features of the early stage of LPLK include hyper granulosis, epidermal hyperplasia, a few necrotic keratinocytes, and a superficial, bandlike lichenoid infiltrate. Clinically these lesions appear as pink macules or papules and may be difficult to distinguish from basal cell carcinoma or squamous cell carcinoma. [20]

Intermediate Stage

Histologically intermediate stage LPLK is characterized by melanophages, inflammatory cells, and fibrosis, with features consistent with either a lentigo or a seborrheic keratosis. In some cases, clinically, the lesion may be difficult to distinguish from melanoma (melanoma on sun-damaged skin, lentiginous melanoma, lentigo malignant melanoma). [20]

Late Stage

Late-stage LPLK is characterized histologically by papillary fibrosis, telangiectasias, and melanophages. The lesions have a more blue-gray to black clinical appearance and may be difficult to distinguish from melanoma. [20]

DF: “Dermatofibroma”

Dermatofibromas (DFs) are prevalent cutaneous lesions that most frequently affect young to middle-aged adults, with a slight predominance in females. Clinically, dermatofibromas appear as firm, single, or multiple papules/nodules with a relatively smooth surface and predilection for the lower extremities. Characteristically, upon lateral compression of the skin surrounding dermatofibromas, the tumors tend to pucker inward producing a dimple-like depression in the overlying skin; a feature known as the dimple or Fitzpatrick’s sign. [21]

Dermatofibroma features:

There are twelve features in Dermatofibroma [Delicate pigment network**,** A central scar-like white patch, Ring-like or donut-shaped globules, White network, Homogenous areas with brown color, Vascular structures, Rarely comedo-like openings, scale, ulceration, peripheral collarette fissures, ridges, and mamillated surface]

Dermatofibroma patterns:

There are eleven patterns in Dermatofibroma [Peripheral delicate pigment network and central white scar-like patch, Delicate pigment network, Peripheral delicate pigment network and central white network, Peripheral delicate pigment network and central homogenous pigmentation, White network, Homogenous pigmentation, White scar-like patch, Multiple focal white scar-like patches, Peripheral homogenous pigmentation, and central white scar-like patch, Peripheral homogenous pigmentation and central white network, Atypical pattern((1) 'Melanoma-like'; (2) 'vascular tumor-lie'; (3) 'basal cell carcinoma-like'; (4) 'collision tumor-like'; (5) 'psoriasis-like')]

VASC: “Vascular lesion”

Cutaneous vascular lesions comprise all skin diseases that originate from or affect the blood or lymphatic vessels, including malignant or benign tumors, malformations, and inflammatory disease. While some vascular lesions are easily diagnosed clinically and dermoscopically, other vascular lesions can be challenging as many of them share similar dermoscopic features. [22]

Blood vessels are critical to the survival and growth of cells and tissues. The rate of growth and ultimate tumor size, whether benign or malignant, is governed at least in part by the tumor’s ability to derive ample blood flow to support the metabolic demands of its cells. Therefore, tumors may manifest signs of an increase in blood vessels and flow. This is readily observed in many basal cell carcinomas (BCCs) in which the clinical morphology often reveals telangiectasias. Although other tumors may not clinically manifest vasculature, vessels are indeed present but may be of small caliber or situated deeper within the skin. Dermoscopy, by providing magnification and visual access to subepidermal structures, has permitted clinicians to observe many of these vessels. [23]

Vascular structures

There are fourteen structure in vascular [Arborizing blood vessels, Milky red globules/areas, Glomerular vessels, Linear irregular vessels, Polymorphous vessels, Corkscrew / tortuous vessels, Crown vessels, Strawberry pattern, String of pearls pattern, Anatomy of normal skin vasculature, Vessels in the tumor microenvironment, Comma vessels, Dotted vessels, Hairpin vessels].

Dataset:

Introduction

As shown above, we have seven types of skin cancer in our dataset and we explained one limitation above, we will show the next limitation that we faced in our dataset.

HAM10000 Dataset:

The HAM10000 dataset, a large collection of multi-sources dermatoscopic images of common pigmented skin lesions, Training of neural networks for automated diagnosis of pigmented skin lesions is hampered by the small size and lack of diversity of available dataset of dermatoscopic images. They tackle this problem by releasing the HAM10000 ("Human against Machine with 10000 training images") dataset. [4]

They collected dermatoscopic images from different populations, acquired and stored by different modalities. The final dataset consists of 10015 dermatoscopic images which can serve as a training set for academic machine learning purposes.

Cases include a representative collection of all-important diagnostic categories in the realm of pigmented lesions: Actinic keratoses and intraepithelial carcinoma / Bowen's disease (apiece), basal cell carcinoma (bcc), benign keratosis-like lesions (solar lentigines / seborrheic keratoses and lichen-planus like keratoses, bkl),

Dermatofibroma (DF), melanoma (Mel), melanocytic nevi (NV), and vascular lesions (angiomas, angiokeratomas, pyogenic granulomas, and hemorrhage, vasc).

More than 50% of lesions are confirmed through pathology, the ground truth for the rest of the cases is either follow-up examination (follow-up), expert consensus (consensus), or confirmation by in-vivo confocal microscopy (confocal). [4]

The dataset includes lesions with multiple images, which can be tracked by the lesion_id column within the HAM10000_metadata file.

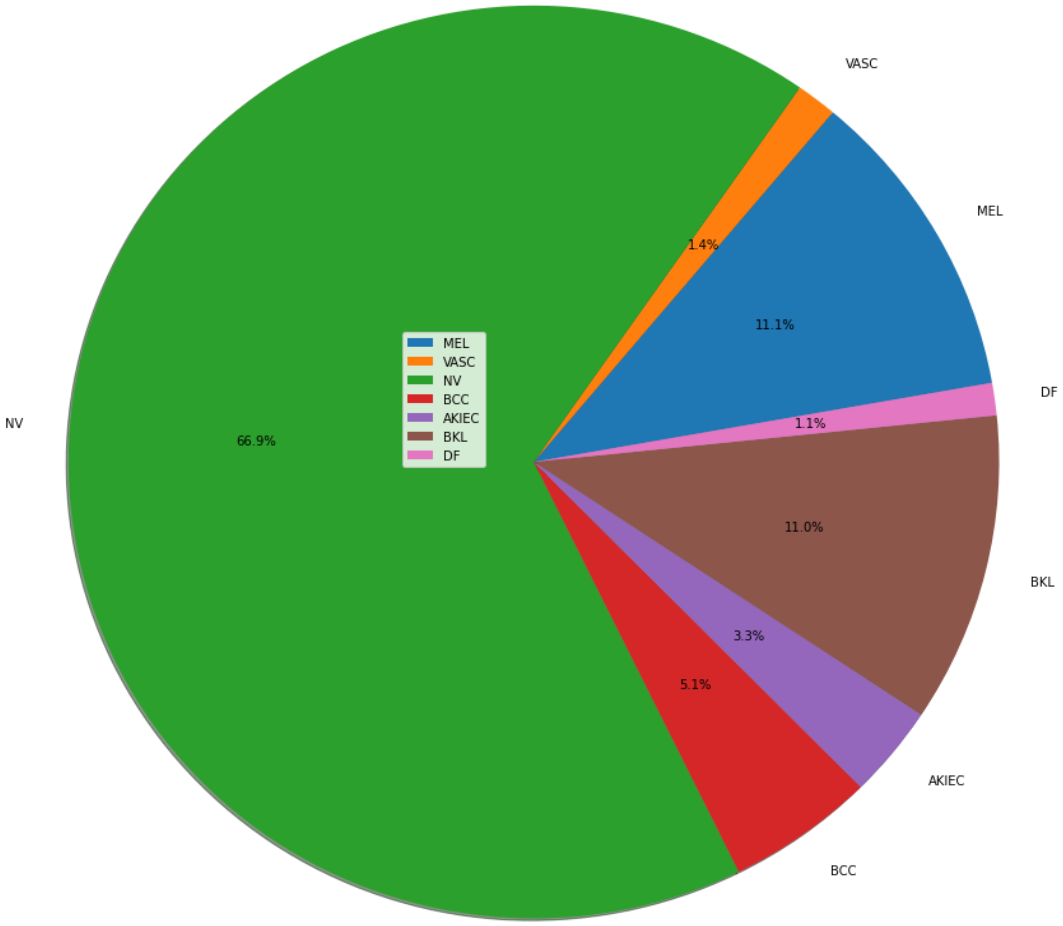
Link of the dataset: https://challenge2018.isic-archive.com/task3/training/

Dataset table

Compare type / Classes	number	percentage	rank by class size
Mel	1113	11.1%	2
NV	6705	66.9%	1
BCC	514	5.1%	4
AKIEC	327	3.3%	5
BKL	1099	11.0%	3
DF	115	1.1%	7
VASC	142	1.4%	6
Classes Sum	10015		

Dataset table (figure 7)

To see how bad these numbers are we draw it as pie chart figure to see the real problem of imbalance, just focus on the number of last 2 classes (df, vasc) both of them are smallest in the figure you will see how small they are compared to others classes specially nv class which is the biggest of them all as we can see in this figure nv is 66.9% from distribution of dataset second biggest class is MEL which is only 11.1% from distribution of a dataset.



Pie chart of dataset distribution (figure 8)

Conclusion:

Every cancer type exist in our dataset has its many unique Dermoscopy features and many structures that maybe have similar features between each other, as well as patterns that change for each cancer and can be exist with others cancers by some kind small probability which make it more difficult to make specific feature extractor algorithm for all cancers with expected high and even if we skip all this each cancer have unfixed number of features that which may affect our training because every single feature has its algorithm with its error and small features that hide in cancer that need too much pre-processing to get without noise.

Sum of error for each cancer feature can be very bad for our training process as well as a different feature has similar things which would take us time to finish each feature extractor which is not effective in our case because we would take time and effort to use digital processing for each case and improve each pointless case and effort waste because of the existence of CNN.

We have an imbalance in our dataset distribution which is not a simple problem that we need to be handled, we have known which methods that we will compare with our experimental methods as a comparison.

Chapter 3 Related work

Introduction:

We will focus on the first three team's approaches that are on the top of the leaderboard of the ISIC task3 challenge.

Teams:

The first is the team called MetaOptima with a balanced accuracy of 88% they used besides the HAM dataset an external data with more than thirty-three thousand photos and from ISIC archive more than four thousand photos. Their approach is they do some pre-processing on the dataset (resize, random square crop, random horizontal flips, random rotations, augment brightness, saturation, and contrasts) then They trained several different models separately on these photos about 19 models and all models their weights are initialized and pre-trained on ImageNet [48]. The reported loss and balanced accuracy are from averaging the results of 5-fold cross-validation. Then Ensemble these models with a stacking scheme [49] this method called Ensembling Convolutional Neural Networks for Skin Cancer Classification [50]

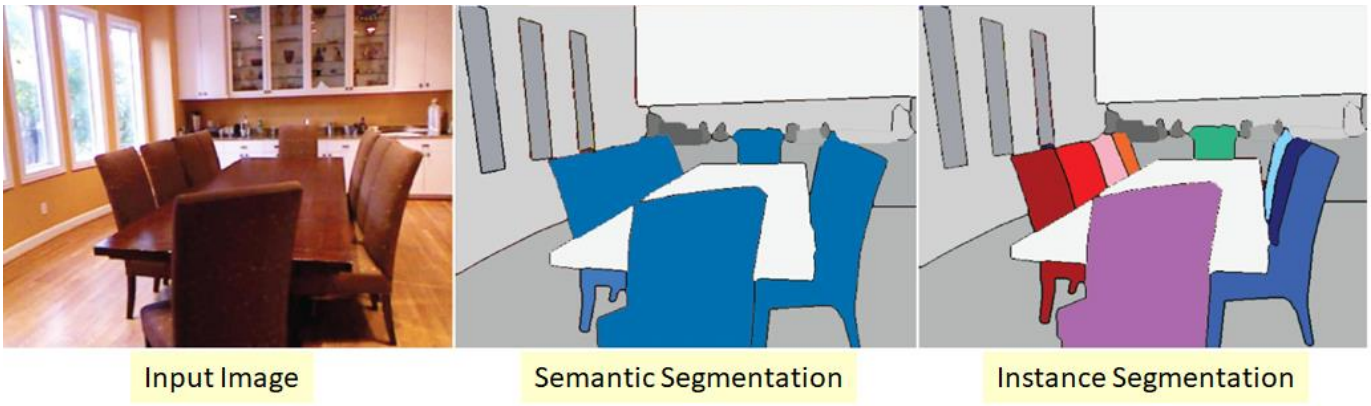
The second team called DAISYLab with a balanced accuracy of 85% they used besides the HAM dataset an external data with more than thirteen thousand photos. Their approach is they do some pre-processing on the dataset (histogram equalization, resize, random cropping, randomly flipped, random changes in brightness and saturation, subtracted the per-channel training set mean) then they trained several different models separately on these photos about 20 models and all models are pre-trained on ImageNet [48]. Then Ensemble these models together and this method called Skin Lesion Diagnosis using Ensembles, Unscaled Multi-Crop Evaluation, and Loss Weighting [51].

The third team called medical analysis group, Sun Yat-Sen University with a balanced accuracy of 84% they didn't use any external data, their approach is they do some pre-processing on the dataset (resize, randomly flipped horizontally and vertically, randomly changed the brightness, contrast, saturation, rotated, affine transformation, and randomly cropped) then they trained two models (Senet and Pnasnet) separately and then use them in the ensemble with test time pre-processing. [52]

Chapter 4 Segmentation

Introduction:

Segmentation is a process that partitions an image into regions. It is an image processing approach that allows us to separate objects and textures in image segmentation is especially preferred in applications such as remote sensing or tumor detection in biomedicine, Image segmentation is considered the most essential medical imaging process as it extracts the region of interest through semi-automatic or automatic process it divides an image into areas based on a specified description, such as segmentation body organs/tissues in the medical application for border detection, tumor detection/segmentation, and mass detection. There are two processes in segmentation semantic segmentation and instance segmentation process, semantic segmentation is the process of assigning a label to every pixel in the image. This is in stark contrast to classification, where a single label is assigned to the entire picture. Semantic segmentation treats multiple objects of the same class as a single entity. On the other hand, instance segmentation treats multiple objects of the same class as distinct individual objects (or instance). Typically, instance segmentation is harder than semantic segmentation [42].



Compare between semantic and instance segmentation (figure 9)

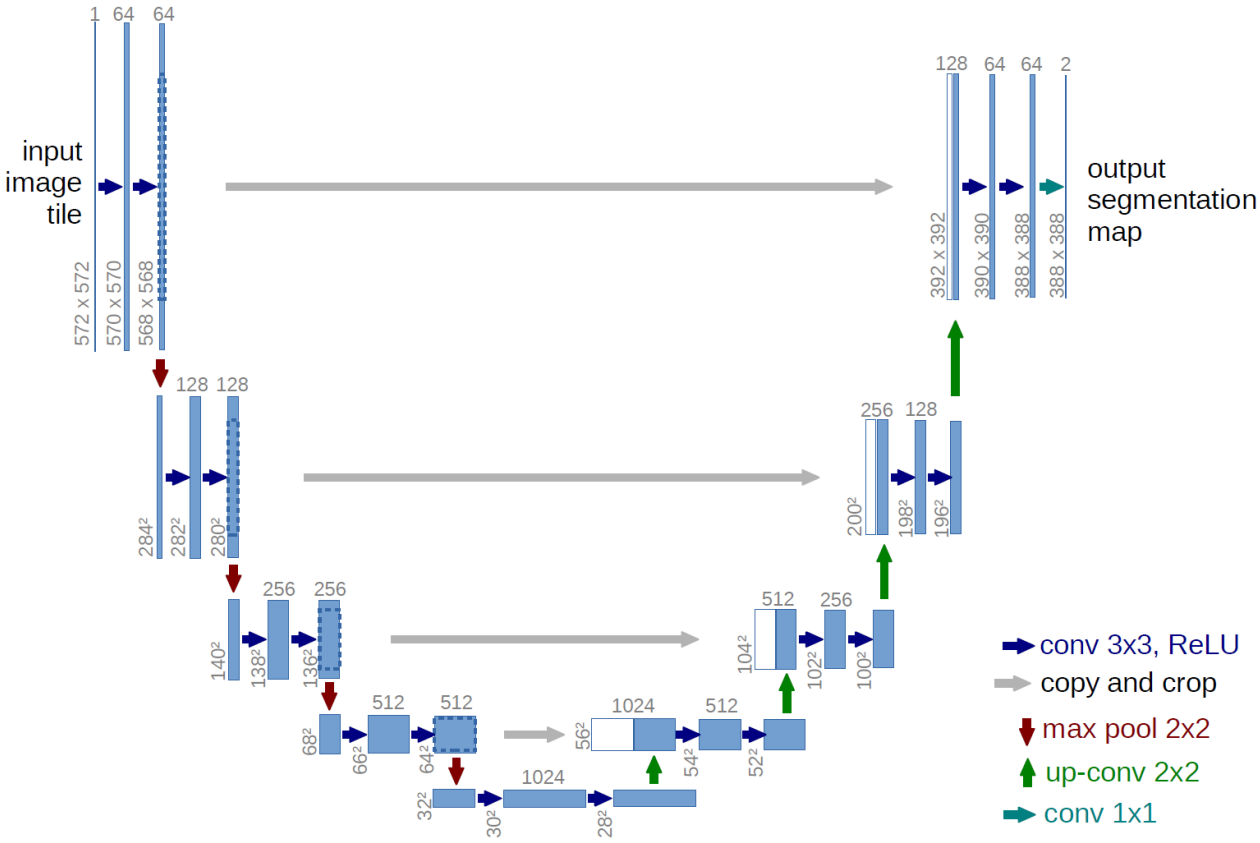
So, we choose semantic segmentation process because we have only two objects one is skin and other is cancer, there are old school images segmentation methods like thresholding, k-means clustering, histogram-based image segmentation, and edge detection, and there are modern image segmentation techniques are powered by deep learning technology and here is a paper that talks about Image Segmentation Using Deep Learning: A Survey [43]. These are the names of some deep learning architectures used for segmentation convolutional neural network, fully convolutional neural network,

ensemble learning, deep-Lab, and U-net. So, we use deep learning technique because we have a supervised dataset contains images with its masks that we can train with but because of the limited images in the dataset we use U-net architecture because it is more successful than convolutional models, in terms of architecture and terms pixel-based image segmentation formed from convolutional neural network layers. It is even effective with limited dataset images. The presentation of this architecture was first realized through the analysis of biomedical images. What makes U-net special is as it is commonly known, the dimension reduction process in the height and width that we apply throughout the convolutional neural network that is, the pooling layer is applied in the form of a dimension increase in the second half of the model.

Architecture:

U-Net: Convolutional Networks for Biomedical Image Segmentation [44], evolved from the traditional convolutional neural network, was first designed and applied in 2015 to process biomedical images. As a general convolutional neural network focuses its task on image classification, where input is an image and output are one label, but in biomedical cases, it requires us not only to distinguish whether there is a disease but also to localize the area of abnormality.

U-Net is dedicated to solving this problem. The reason it is able to localize and distinguish borders is by doing classification on every pixel, so the input and output share the same size. For example, for an input image of size 2x2, the output will have the same size of 2x2.



U-net architecture (figure 10)

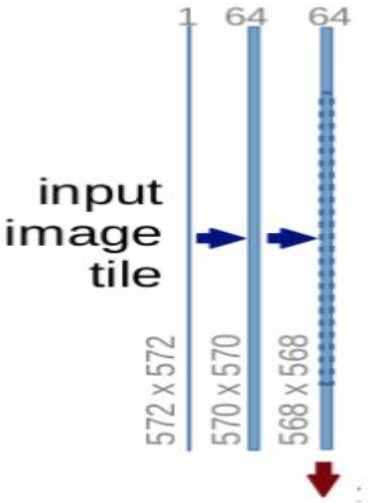
U-net takes its name from the architecture which visualized, appears similar to the letter U, as shown in the figure above. Input images are obtained as a segmented output map. consists of two major parts — the left part is called contracting path, which is constituted by the general convolutional process; the right part is an expansive path, which is constituted by transposed 2d convolutional layers, The most special aspect of the architecture in the second half. The network does not have a fully connected layer. Only the convolutional layers are used. Each standard convolutional process is activated by Re-LU activation function.

Implementation:

First, let's talk about Contracting Path

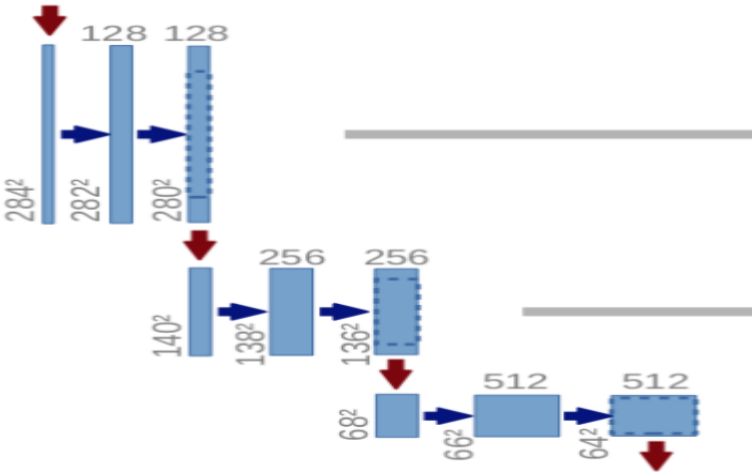
The contracting path follows the formula:

conv_layer1 -> conv_layer2 -> max pooling -> dropout (optional) (layers)



Input layers (figure 11)

Each process constitutes two convolutional layers, and the number of channel changes from 1 → 64, as the convolution process will increase the depth of the image. The red arrow pointing down is the max pooling process which halves downsize of an image (the size reduced from 572x572 → 568x568 is due to padding issues, but the implementation here uses padding= “same”).
The process is repeated 3 more times:



Left layers (figure 12)

And now we reach at the bottommost:

Bottommost layers (figure 13)

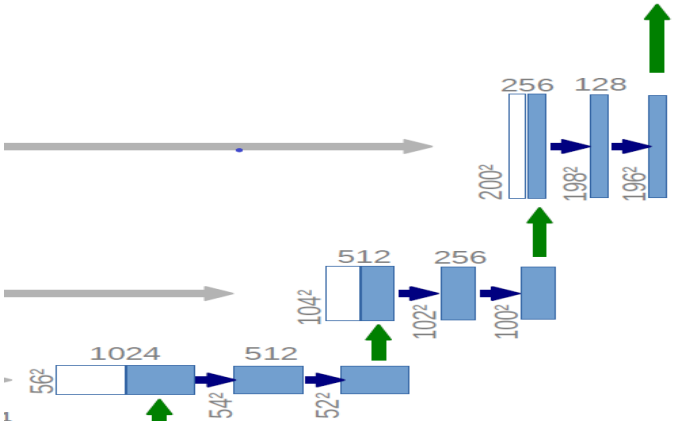
Still, 2 convolutional layers are built, but with no max-pooling:

The image at this moment has been resized to 28x28x1024. Now let's get to the expansive path.

Expansive Path:

In the expansive path, the image is going to be upsized to its original size. The formula follows:

conv_2d_transpose -> concatenate -> conv_layer1 -> conv_layer2 (layers)



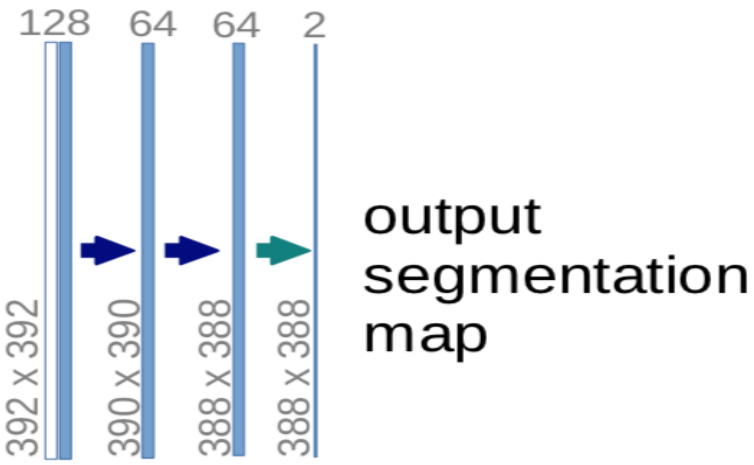
Right layers (figure 14)

Transposed convolution is an up-sampling technique that expands the size of images. There are a visualized demo and an explanation in the references. Basically, it does some padding on the original image followed by a convolution operation.

After the transposed convolution, the image is upsized from 28x28x1024 → 56x56x512, and then, this image is concatenated with the corresponding image from the contracting path and together makes an image of size 56x56x1024. The reason here is to combine the information from the previous layers in order to get a more precise prediction.

Same as before, this process is repeated 3 more times:

Now we've reached the uppermost of the architecture, the last step is to reshape the image to satisfy our prediction requirements.



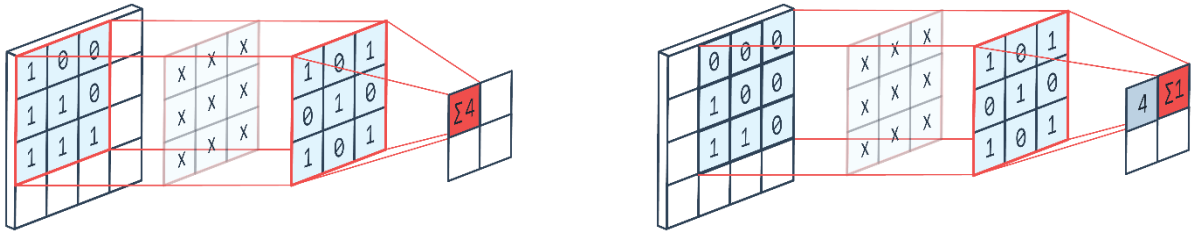
Output layers (figure 15)

The last layer is a convolution layer with 1 filter of size 1x1(notice that there is no dense layer in the whole network). And the rest left is the same for neural network training.

Layers

Convolution

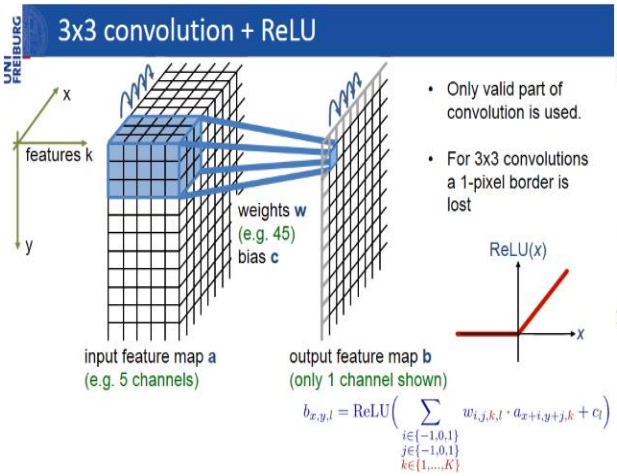
The 2D Convolution block represents a layer that can be used to detect spatial features in an image, either working directly on the image data or on the output of previous convolution blocks.



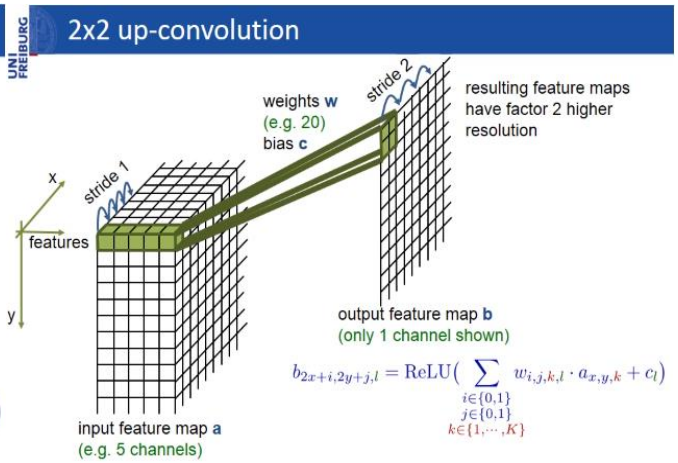
2D Convolution (figure 16)

Each layer is composed by a configurable number of filters, where each filter is a HxWxC matrix of trainable weights; a convolution operation is performed between the image and each filter, producing as output a new image (tensor in DL) with height and weight determined by the input image, stride and padding (the output height and weight are inversely proportional to the stride) and as many channels as the number of filters. Every value in the tensor is then fed through an activation function to introduce nonlinearity.
Each pixel in the image represents how strongly the corresponding feature (which could be an edge, a color gradient in the original image, or a certain configuration of edges in a deeper layer of the network) is present in the HxW area centered on that pixel.
This is a nice example of a filter that can detect edges; while historically computer vision and image processing relied on fixed shape feature detectors, convolutional neural networks learn the best filter shapes for the task at hand.
Every layer has (filter width) x (filter height) x (number of channels in the layer input) x (number of filters) weights. This is typically much smaller than the size of the input image, unlike what would happen for dense layers (where each pixel would get a weight for every neuron).
The default is to move filters by 1 pixel at a time when performing convolutions; this is called stride and it can be altered by the user. The bigger the stride, the smaller the output image will be along the corresponding axis. This can be used to reduce the number of parameters and memory used but leads to a loss of resolution.

These layers are intended to increase the resolution of the output. For localization, the sampled output is combined with a high-resolution feature throughout the model. A sequential convolutional layer aims to produce a more precise output based on this information.



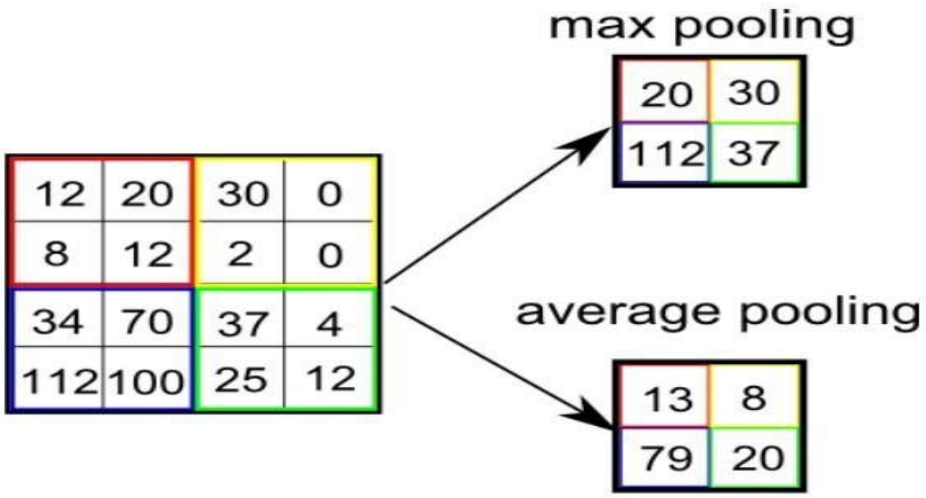
3x3 convolution + ReLU (figure 17)



2x2 up-convolution (figure 18)

Pooling layers

The pooling layers reduce height and width information by keeping the number of the input matrix constant. The calculation is a step used to reduce complexity (each element of the image matrix is called a pixel) in summary, the pooling layer refers to a pixel that represents a group of pixels Note: pooling layers can work with different approaches including maximum, average, or median layers and for more explanation for convolution layer and transpose convolution layer [45]



Pooling layer (figure 19)

The pixels in the border region are symmetrically added around the image so that image can be segmented continuously. With this strategy, the image is segmented completely. The padding method is important for applying the U-net model to large images; otherwise, the resolution will be limited by the capacity of the GPU memory. And we choose for loss approach is binary cross-entropy because there is only one class we interest in and it is the cancer.

$$L_{bce} = \sum_i y_i \log o_i + (1 - y_i) \log (1 - o_i)$$

Binary cross entropy loss function (figure 20)

So, in conclusion, U-Net is able to do image localization by predicting the image pixel by pixel and the network is strong enough to do good prediction based on even few data sets by using excessive data pre-processing techniques so that's why we choose this architecture to use on skin cancer dataset ISIC task3 challenge [46] this dataset has 2075 pictures we split it into 1867 train and 208 validate and 519 for test use it to evaluate the trained model we use this dataset to train the model to predict the mask that contains the infected area.

So, after explain why choosing the segmentation with a deep learning architecture U-net with a loss function binary cross-entropy comes the training process.

Training process:

We first used a pre-processing method on the dataset train and validate before passing it to the model to train.

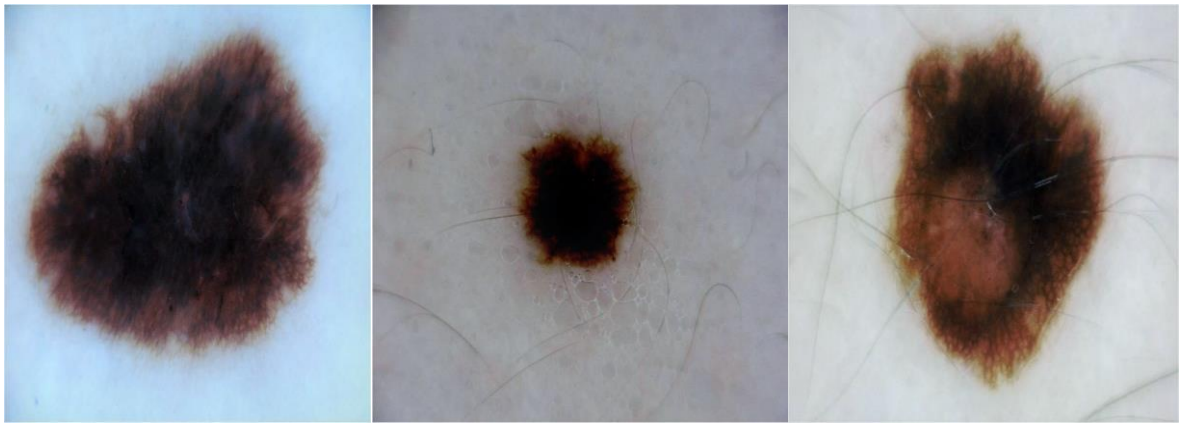
First, we used the auto color equalization (ACE) algorithm this algorithm is able to adapt to widely varying lighting conditions and able to extract visual information from the environment efficaciously. ACE has shown promising results in solving the color constancy problem and performing an image dynamic data-driven stretching [47] by using this algorithm it made the object in our case cancer more visible and denser in its colors as we can see the difference in the pictures below from training images.

Pre-processing:

Before pre-processing:

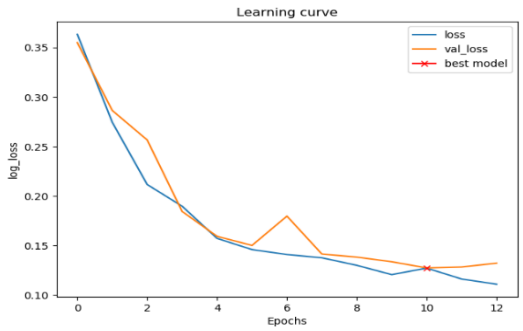
Before pre-processing (figure 21)

After pre-processing:

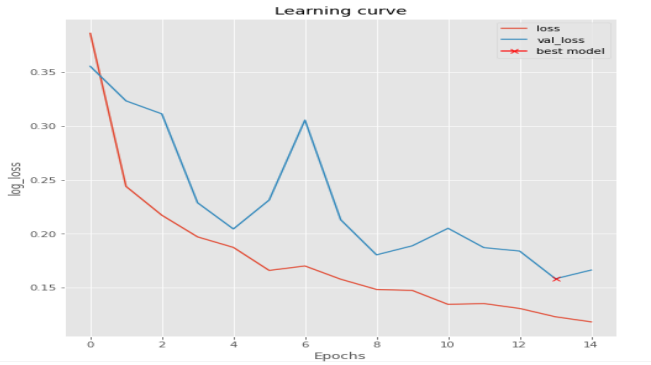


After pre-processing (figure 22)

And by using this pictures to train our model it increased its accuracy then resize the images with its masks to the size of (128,128) then pass it to the model to train so our trained model 1 with auto color equalization achieved an accuracy of 93% within 13 epoch but if we trained or model 2 with the data without the ACE within 15 epoch it achieved 90% here this figure represents training and validation loss of model that trained when applying ACE on images and without as we can see that the figure that has staple training and validation loss belong to the model who trained on preprocessed images.



Learning curve for model 1 (figure 23)



Learning curve for model 2 (figure 24)

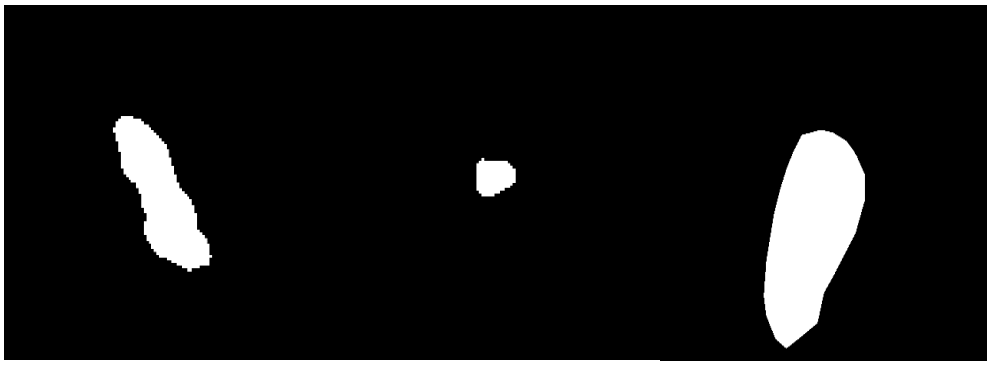
Model results:

By using model 1 and model 2 we got some predicted masks we can compare it with the real masks from the test picture below.

Original pictures:

Original pictures (figure 25)

Original masks:

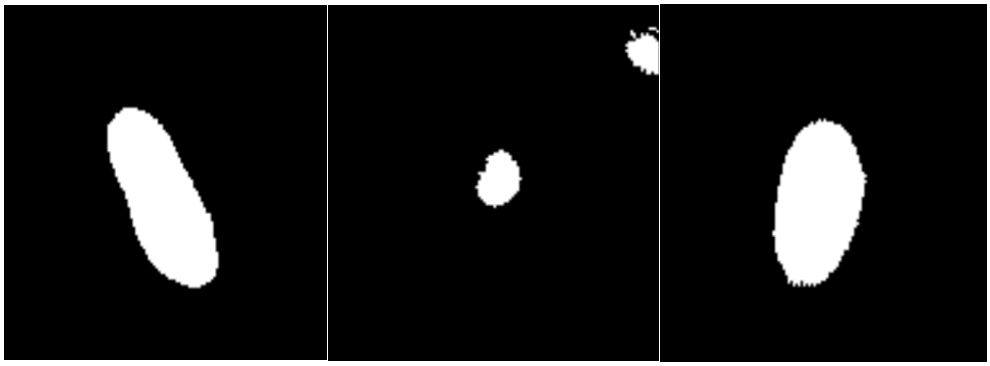


Original masks(figure 26)

Model1 Predicted masks:

Model1 Predicted masks (figure 27)

Model2 Predicted masks:



Model2 Predicted masks (figure 28)

As we can see that model 1 is better than model 2 in the prediction of masks because it can contain more of the infected area than model 2 so by using the trained model 1, we can detect cancer with more accuracy than the other model and make a mask of the detected cancer area and use it to either segment this infected area or use this predicted mask to use it with our method called limited certain crop.

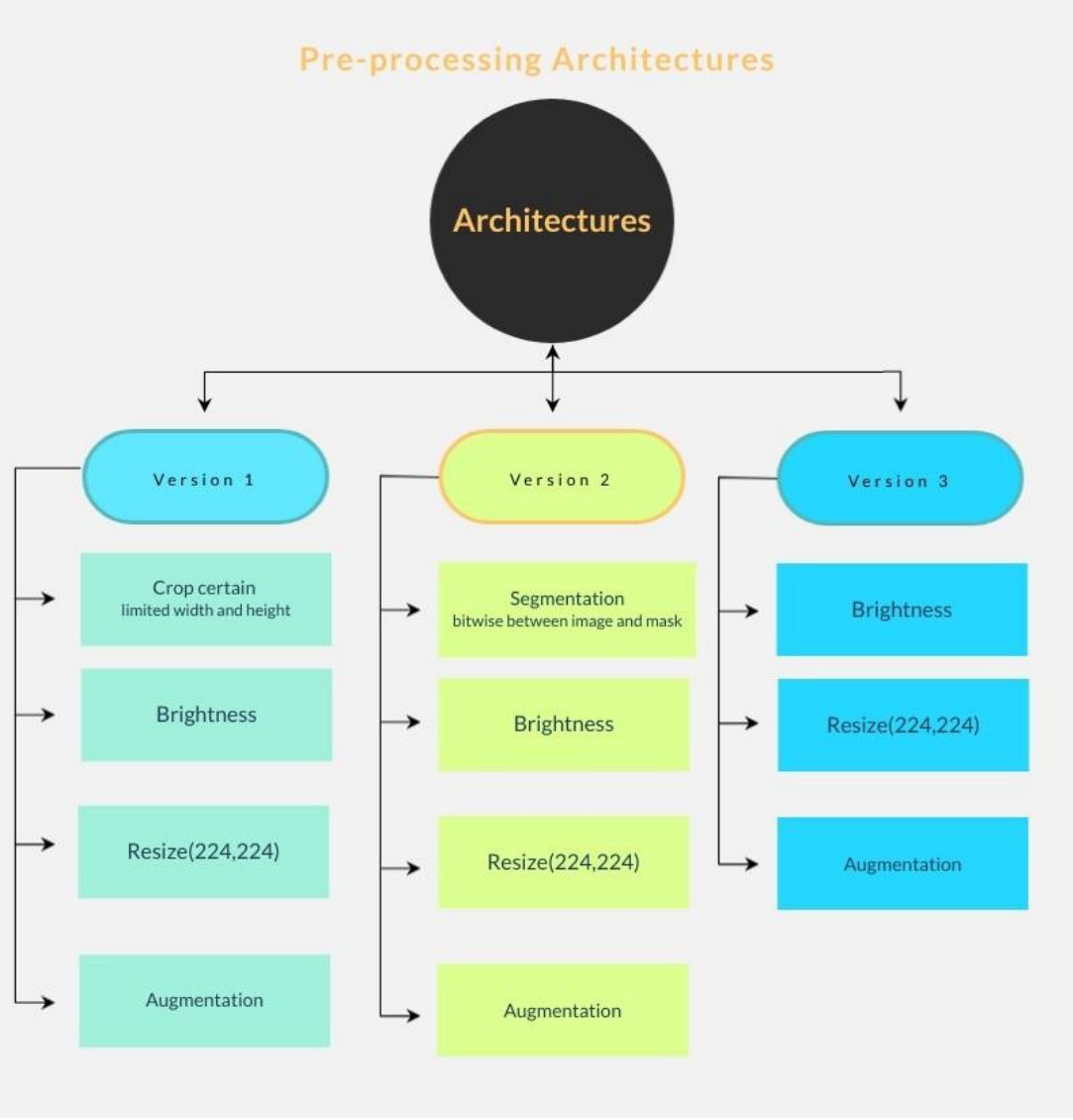
Conclusion:

U-net showed that is one of best choices in medical segmentation and our first segmentation model with this Architecture show more promising result than our second segmentation model which show the impact of color equalization on our training data and test data of this segmentation model.

Chapter 5 pre-processing

Introduction

We have three main architectures for each main one has similarity with other that we will talk in part called architectures similarity but each architecture has its own unique pre-processing, this chapter will explain each one and its advantages and disadvantages from our point of view.



Pre-processing architecture (figure 29)

As we can see in this figure brightness and resize and some pre-processing are in architectures similarity part, we will explain them as a unit at the end of this chapter

We will talk about each version and reason for making this version with its unique parts.

Version 1

Reason:

solution for crop certain defects that we predicted and it affect our results in baseline models so we made this function as a solution and to add 10 pixels in each line of the box to add part of the skin as error ratio and bigger images will not need error rate or limited crop certain to be applied because the more size came from crop certain the more probability of cancer to exist in the box that came from crop certain which will solve both resize problem and error problem of the segmentation model.

Unique pre-processing:

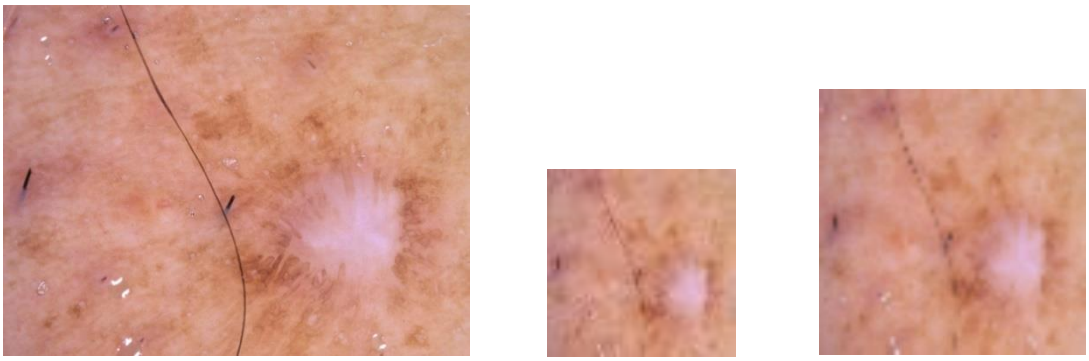
There is only one unique pre-processing which is Limited Crop certain

Limited Crop certain: is one of our experimental method that was a solution for crop certain noise by the limit box that contains cancer to be at least at the size of our resize method size (224,224) plus the size of the error that if we want to add it we will sum it with resizing in our case was 10 pixels for each min and max of width which make total is 224+20=244 then it would be no problem with resizing bigger image as far we would explain below.

So why should we do that in this case, we can just pass the images as it is after crop certain but the answer is in resize method itself and how it works and how image scale works because we do not have fear about images that have height and width that bigger than resize method size it will just lose some rows or columns to fit the size without adding any pixels, we do worry about what may happen if we resize the image that is smaller than the resize size

Example:

We have an image that we resize it to be like one of the smallest images that will come out from crop certain.

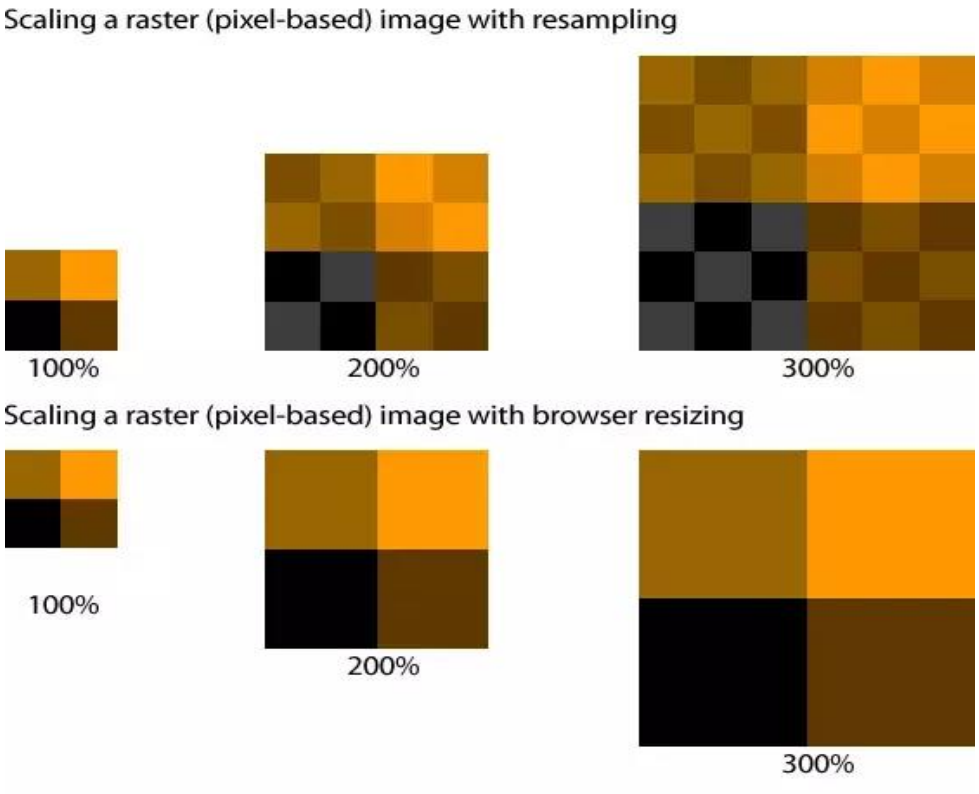


Original picture → Resize to (60*60) → Resize to (224*224)

Resizing examples (figure 30)

As we can see resize small images to bigger ones will make it pixelated and add more noise to image and this because that resize add pixels to resize or scaling up images could be replicate to neighbors' pixels or interpolate and the smaller the image the more effect of pixelated

To understand what we mean we find a figure that proves our point of view



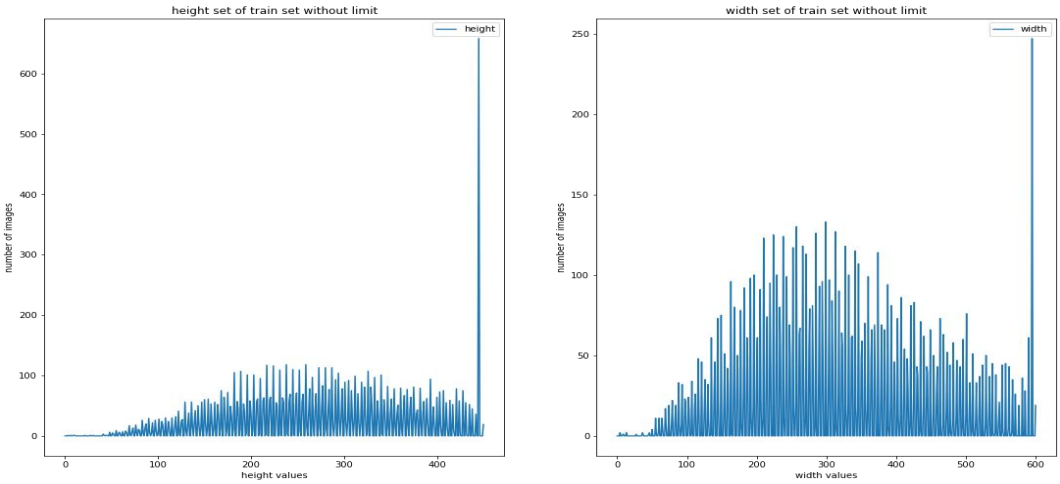
Scaling technique example (figure 31)

As we see in both cases it will add pixels which in our point of view counted as noise and because there is images are even smaller than 60*60 came out from crop certain

method not only in width or only height but both so we made some analysis on our dataset and test set to compare both width and height in each image compared to the size of resize (224,224) with figures that can show the distribution of each size.

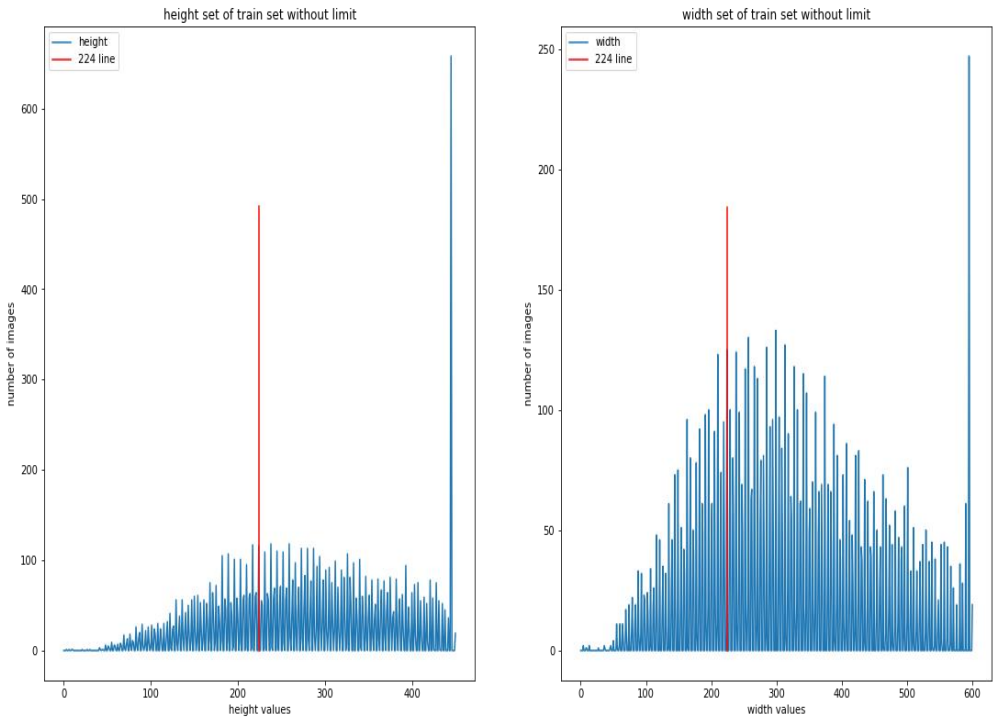
Analysis:

By converting the size of images into plot figures that will explain our point of view



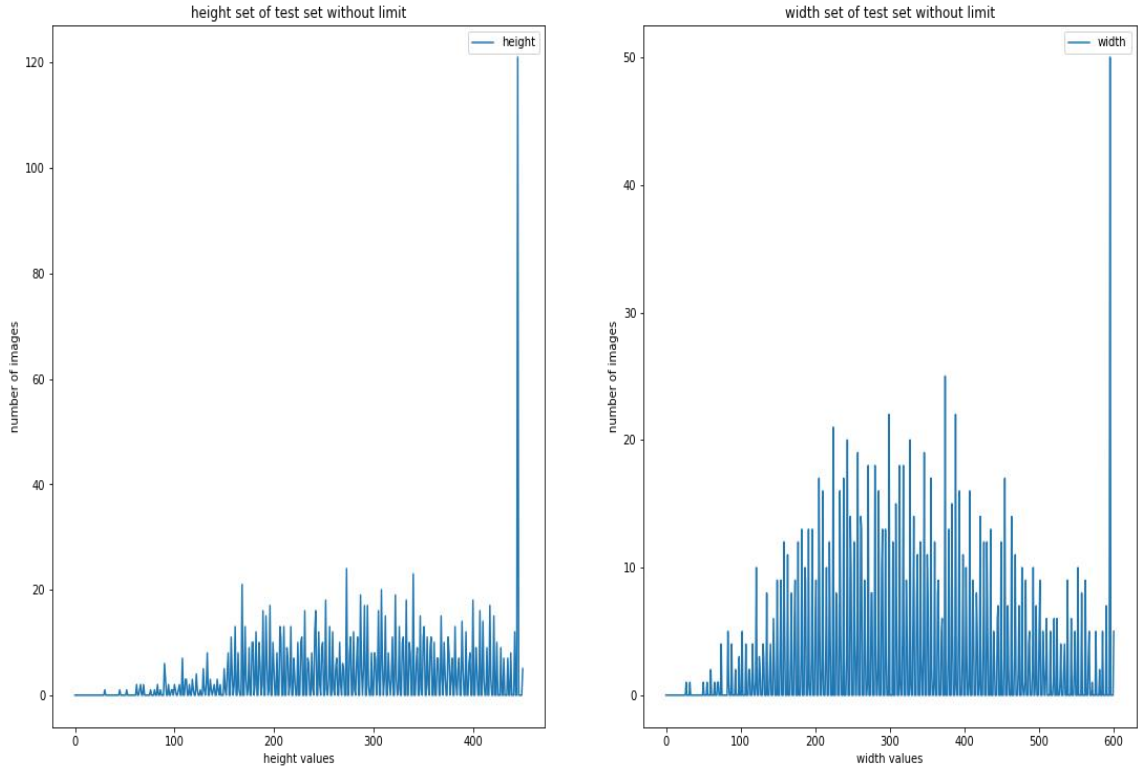
Cropping train set plot without limit (figure 32)

As we can see height and width that came out from crop certain let's see how many images, we lose due resize with 224



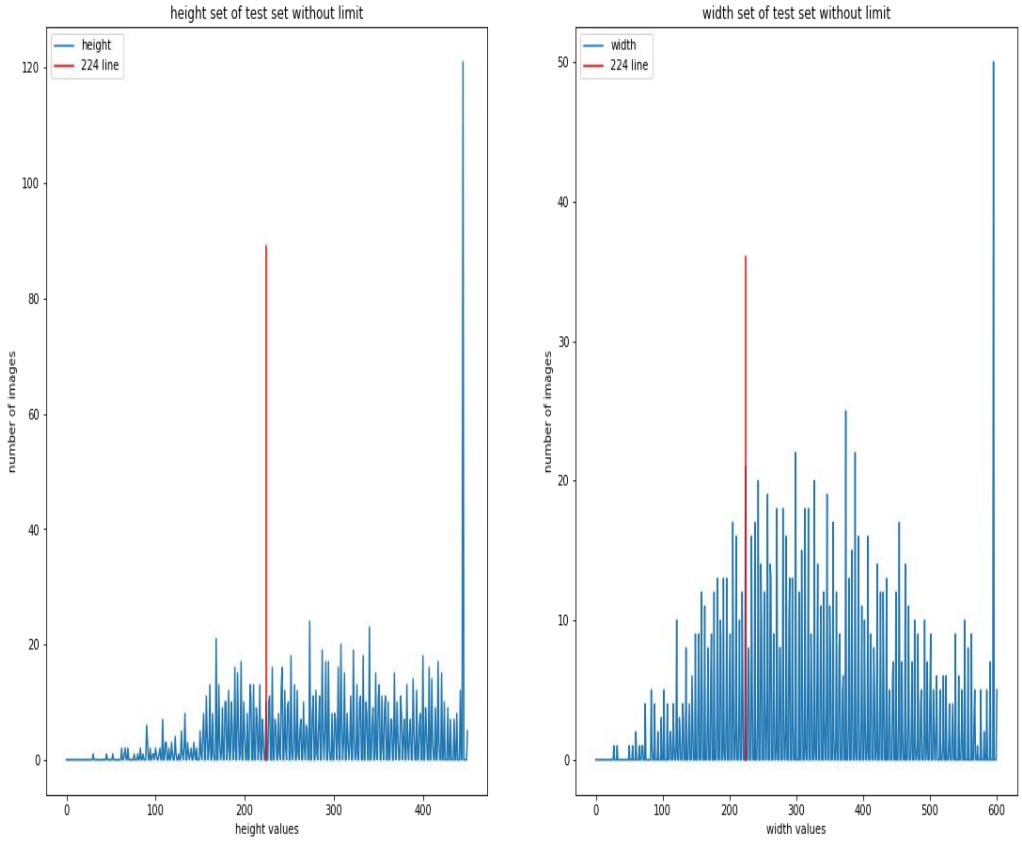
Cropping train set plot without limit with compare line (figure 33)

We lose over 3220 images in width and 3059 images in height in the train set, the total loss in train width height by removing repeated images are 3880 images that will have this problem so 3880 pixelated images and smaller the worsen So we have at least 38% from images in the train that will have this problem, let's see how many images in the test



Cropping test set plot without limit (figure 34)

As we can see height and width distribution in test let's see how many did, we really lose in test



Cropping test set plot without limit with compare line (figure 35)

We lose over 427 images in width and lose over 400 images in height in the test set, the total loss in test width height by removing repeated images are 515 images that will have this problem so 515 pixelated images and smaller the worsen

So, we have at least 34% from images in the test that will have this problem

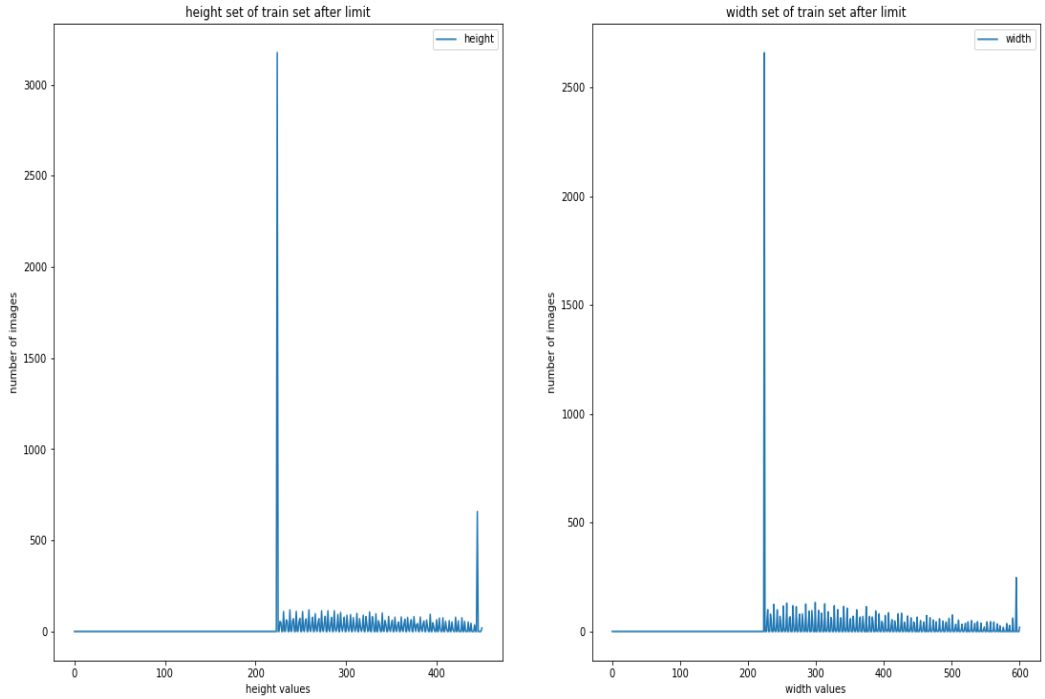
Small images will make the situation worse but how about double damage that means the image has both width and height that less than 224 so we calculated single damage and double damage

Single damage only height or only width in the train are 660 images in height and 821 images in width with a total of 1481 images would suffer from a single damage

Single damage only height or only width in the test are 88 images in height and 115 images in width with a total of 203 images would suffer from a single damage

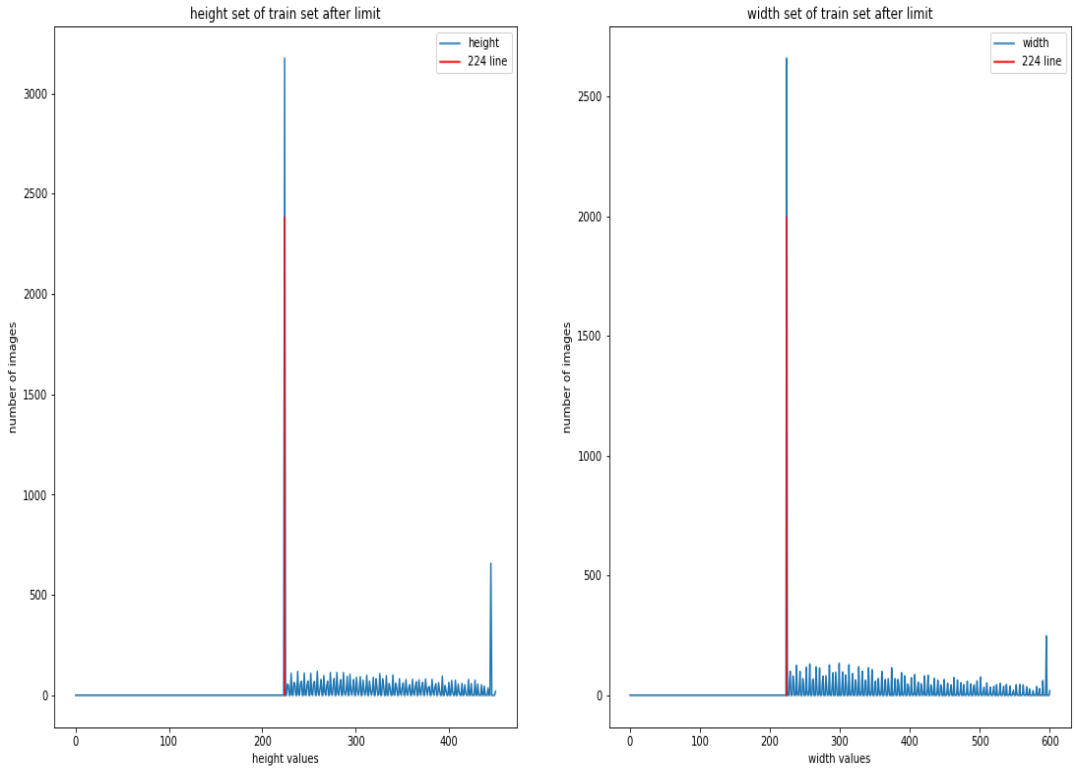
Which leave double damage of width and height in the train to be 2399 images and leave double damage width and height in test to be 312 images

This way we use limited crop certain because it will move images below 224 sizes in both width and height to the right place.



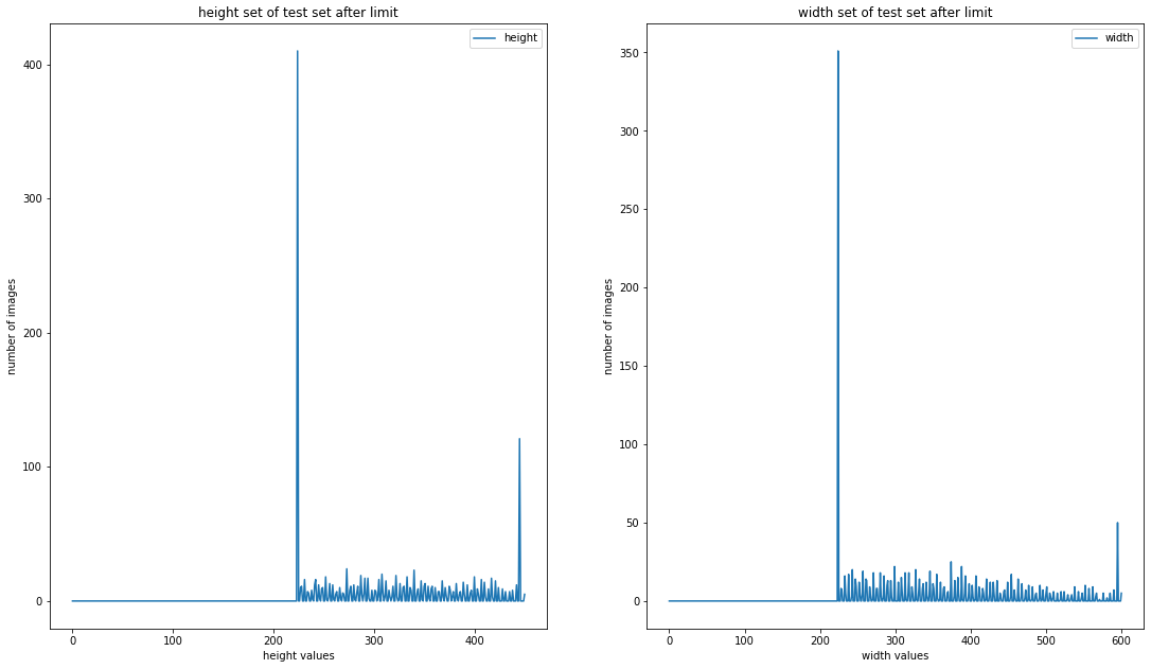
Cropping train set plot with limit (figure 36)

As we can see in this figure most of the data go to 224 value which makes this method as zoom for small images and crops certain for bigger images as we would see in next figure but it is better than random neighbor added to pixels of an image because sometimes edit the image itself and delete some of the features and replace it with others pixels.



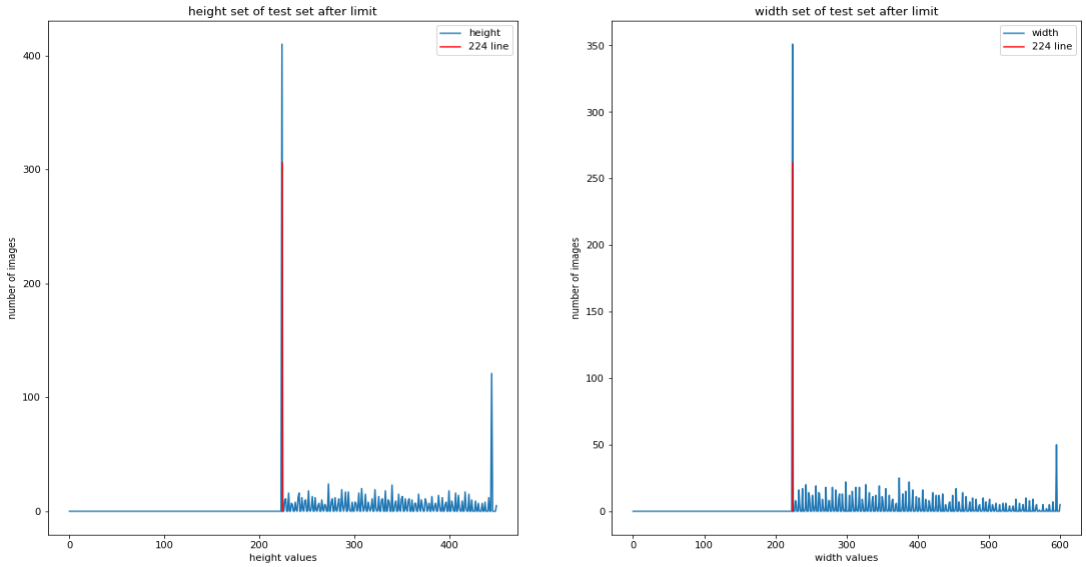
Cropping train set plot with limit with compare line (figure 37)

Same with test all damaged images moved to 224 size



Cropping test set plot with limit (figure 38)

As we can see in the next figure that all damaged images are already on the red line



Cropping test set plot with limit with compare line (figure 39)

Let's see smallest images in train and test and how really small they are before our method we sum both width and height of each image then we make array of first 10 smallest images in both train and test

First smallest 10 images in train

['ISIC_0027249': 7, 'ISIC_0025362': 13, 'ISIC_0027181': 14, 'ISIC_0025339': 19, 'ISIC_0031345': 34, 'ISIC_0028453': 40, 'ISIC_0027162': 57, 'ISIC_0028400': 77, 'ISIC_0029501': 85, 'ISIC_0028247': 91]

First smallest 10 images in test

[{'ISIC_0035417': 62, 'ISIC_0036022': 89, 'ISIC_0034583': 95, 'ISIC_0035005': 107, 'ISIC_0034914': 122, 'ISIC_0034725': 129, 'ISIC_0035573': 134, 'ISIC_0035245': 134, 'ISIC_0034561': 140, 'ISIC_0035272': 140]

Smallest image in train:

Id=ISIC_0027249

Width=4

Height=3

As we can see ISIC_0027249 is too small to be resized to 224 * 224 in train set

Smallest image in test:

Id=ISIC_0035417

Width=32

Height=30

As we can see ISIC_0035417 is small to be resized to 224 * 224 not as smallest one in train but it still very small

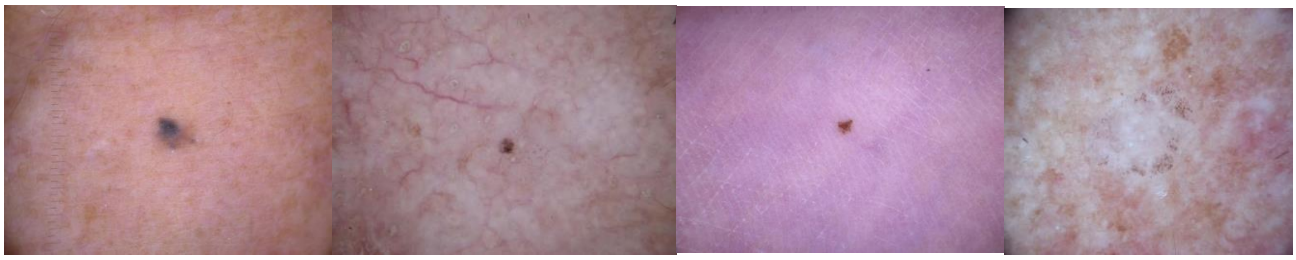
As we can notice both train and test have images that are far smaller than our example 60*60 at the beginning which proves our point.

Examples:

We will show the number of images after applying the method from the smallest images from train and test

Real images:

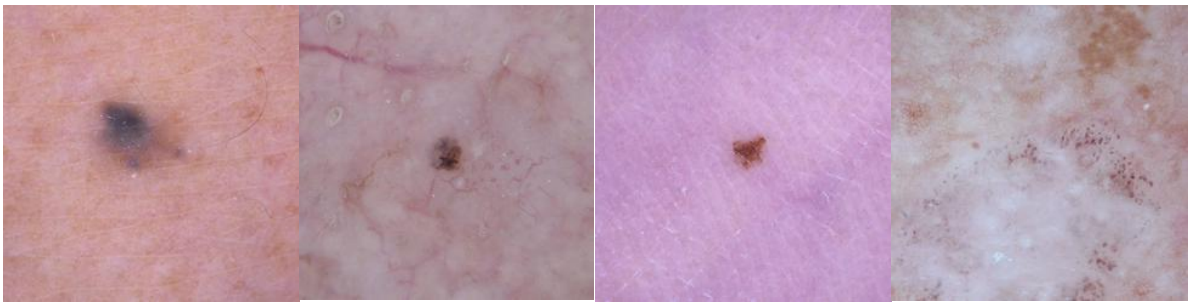
As we can see those images with small cancer in each one of them.



Train1 ISIC_0027249 Train2 ISIC_0025362 Test1 ISIC_0035417 Test1 ISIC_0035417

Small cancer examples before limited crop certain method (figure 40)

After the method:



Train1 ISIC_0027249 Train2 ISIC_0025362 Test1 ISIC_0035417 Test1 ISIC_0035417

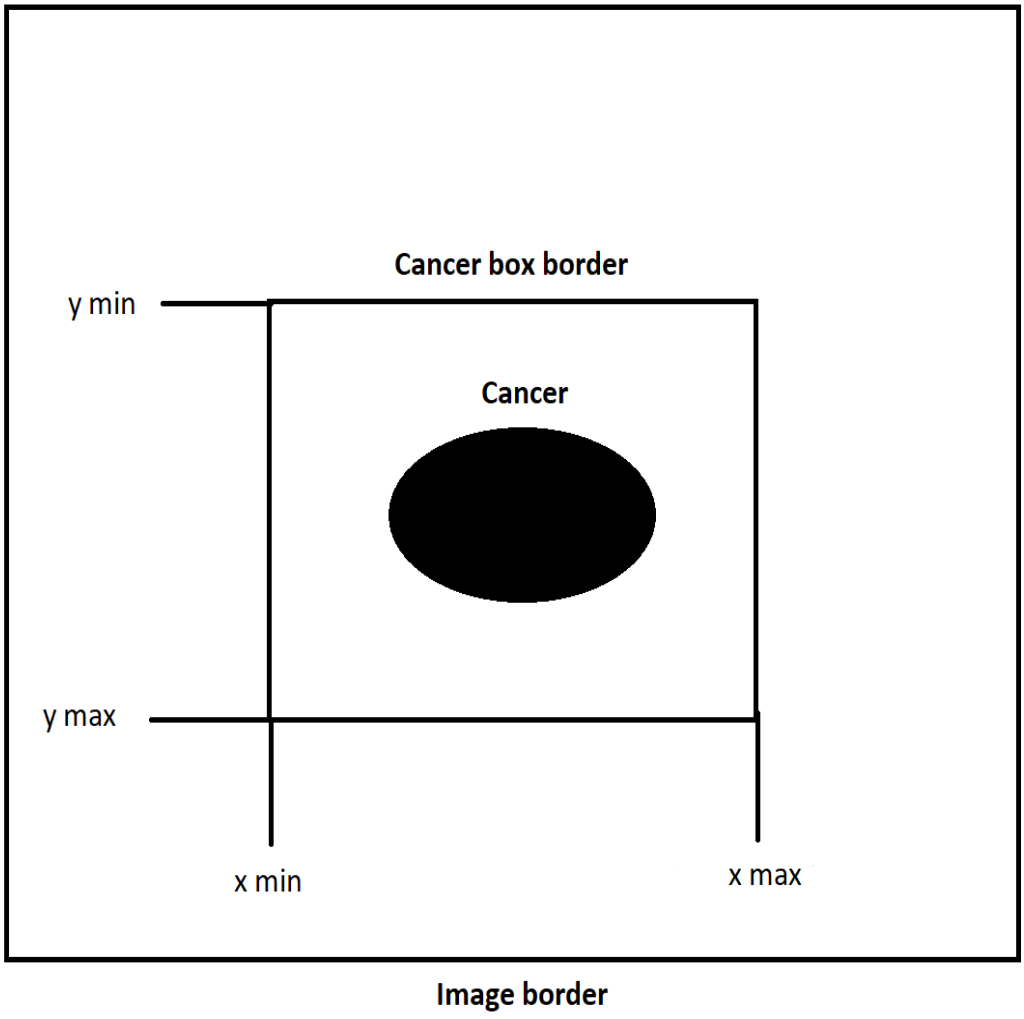
Small cancer examples after limited crop certain method (figure 41)

As far we explained why we chose it, we still did not explain how it really works so we have to far explain how it works then give pseudocode example.

How it works

Our algorithm will only work with images that smaller than resize value in our case it is 224+20 which make our algorithm zoom for those images that are smaller in either width (x) or height (y) and for those which are bigger will ignore them to be normal crop certain, the algorithm works with one in each time which mean it will work only for min-width and max-width or only for min-height and max-height.

then Our algorithm will take the space between x or y min and x or y max then subtracted it from resizing to gain the space that needs to increase it for max x or y and space to decrease from x or y min and this space will be half for min and a half to the max but if one of min or max values hit the border which is zero for min and given to algorithm in case of max will add the remaining space of the space that should have been increased or decreased into the other edge, at last, it returns an array from two numbers, the first number represents new x or y min and the second number represent new x or y max after we applied that algorithm for both widths (min, max) then height(min, max) we will have the new points of a box that contain cancer.



x/y-min and x/y max explanation (figure 42)

Limited crop certain pseudocode:

Algorithm: Limiter

input : xy-box-min , xy-box-max , xy-image-border , resize
output: xy-box-min-new , xy-box-max-new

Begin;
xy-box-min-new = Zero;
xy-box-max-new = Zero;
if *(xy-box-max) − (xy-box-min) >= resize* **then**
  Return[(xy-box-min),(xy-box-max)];
end
space = resize − ((xy-box-max) − (xy-box-min));
half = ceil(space/2);
border-max-space = (xy-image-border) − (xy-box-max);
if *border-max-space >= half* **then**
  xy-box-max-new = half + xy-box-max;
else
  xy-box-max-new = xy-box-max + border-max-space;
  half = ($(2 \times half)$ − border-max-space);
end
if *xy-box-min > = half* **then**
  xy-box-min-new = (xy-box-min) − half;
else
  xy-box-max-new = xy-box-max-new + (half − xy-box-min);
end
Return[(xy-box-min-new),(xy-box-max-new)];

Limited crop certain pseudocode (figure 43)

Advantages:

1-) This method won't just delete the expected noise from resizing but will help in minimizing the error that came from segmentation model itself because in case of the wrong prediction it may guarantee that part cancer will be in the image in case of bigger cancer but in case of small cancer size 224 *224 is 50% height *37% width compared to the biggest image in the dataset which is a few in dataset compare to others sizes which are acceptable in our point of view by looking in our height and width analysis figures

2-) ability to add part of skin or error space to cancer by sum number of pixels to wanted resize that taken as input in the algorithm

Disadvantages:

1-) it is fixed to the size of resizing, so if cancer is too small it will count as the center zoom on cancer that in the center not as segmentation (crop certain)

2-) border of min or max will increase by half of the need of space if it hit the border of either max or min it will increase only max or decrease the only min to increase the subtract of min and max to be 224+error space so if cancer was in borderline, it will add more skin to the output image.

Version 2

Reason:

We made this version so we would be able to compare our method limited crop certain to this version which is counted as normal segmentation

Unique pre-processing:

There is only one unique pre-processing which is the bitwise image with its mask to show only features of cancer and delete the rest of the skin

Examples:

We will show the number of images after applying the method from the smallest images from train and test

Real images:

Train1 ISIC_0027249 Train2 ISIC_0025362 Test1 ISIC_0035417 Test1 ISIC_0035417

Small cancer examples before bitwise method (figure 44)

After the method:

Train1 ISIC_0027249 Train2 ISIC_0025362 Test1 ISIC_0035417 Test1 ISIC_0035417

Small cancer examples after bitwise method (figure 45)

Advantages:

1-) it will show only all features of cancer in case of right prediction which is the focus of our model by putting a black background on all image and leave cancer features only

2-) it will skip the problem of resizing in crop certain

Disadvantages:

1-) if there is the wrong prediction on image segmentation it would destroy most cancer features and leave only skin features

2-) sometimes showing part of the skin that is near to cancer is useful in diagnoses but bitwise only can show features of cancer that were predicated

Version 3

Reason:

Sometimes segmentation would have a small effect on the dataset so people use different pre-processing to replace it, so we choose to zoom to replace it. Of course, there are many methods but the first methods to replace segmentation on cancer it would be zoom or crop

Unique pre-processing:

There is only one unique pre-processing which is zoom one of the pre-processing s that make random zoom with a maximum zoom of 1.5x at the image.

Examples:

We will show the number of images after applying the method from the smallest images from train and test

Real images:

Train1 ISIC_0027249 Test1 ISIC_0035417

Small cancer examples before random zoom method (figure 46)

After the method:

Because this method is random, we should run it multiple times

Train1 ISIC_0027249

Test1 ISIC_0035417

Small cancer examples after applying random zoom method multiple times (figure 47)

Advantages:

It replaces the segmentation model

Disadvantages:

It is not as accurate as the segmentation model

Architectures similarity

Introduction

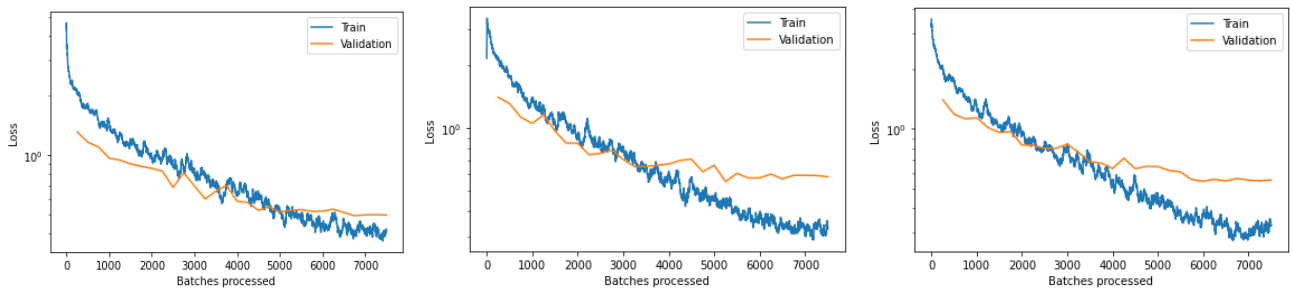
In this section, we will explain the repeated methods in each architecture, and each repeated pre-processing like random brightness, resize random warp, and random rotate.

And we will give an example for each pre-processing and every pre-processing, we will explain if there is a reason behind our choice or we just add it as pre-processing for our models.

Random Brightness:

We wanted to do color constancy in our dataset so we tested some of the methods like the stretch and automatic color equalization then we compare it with random brightness by range (0.5) which was our first try then we tested how its impact on the test set if we change the brightness by range to smaller 0.4 or bigger 0.6 will discuss all this and every method and its impact on the test set and we will explain why did we choose brightness and why we choose a range of 0.5 in brightness.

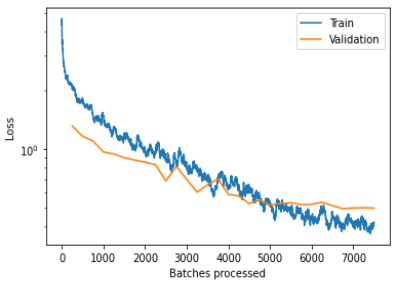
In the beginning, we choose random CNN model (densnet 201) and make all hyperparameters fixed as well as pre-processing s and we compare stretch, color equalization and brightness of range 0.5



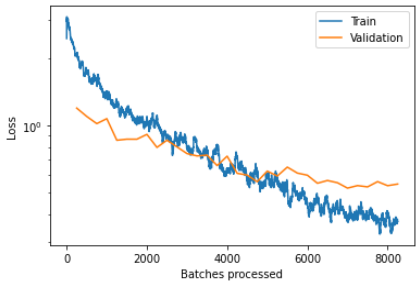
Brightness 0.5 (test score 78) stretch (test score 74) color equalization (test score 72)

Compare loss over batches plots between methods (figure 48)

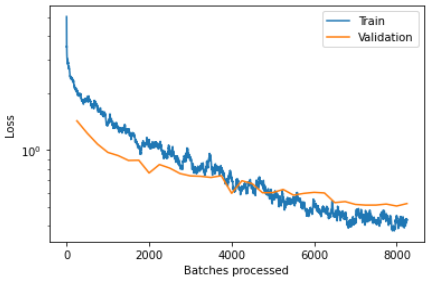
As we can see Brightness did more score as an impact on the test and more stability in learning more than stretch and color equalization this why we favorite brightness over them then we did choose range 0.5 as random so we had to test the other ranges



Brightness 0.5 (test score 78)



Brightness 0.4 (test score 76)



Brightness 0.6 (test score 74)

Loss over batches plots for brightness with different values (figure 49)

We chose a different approach for the train than common ones that why do not mind how Brightness 0.6 more stable yet lower than Brightness 0.4 we will far try to explain why this behavior happens but no Doubt about test score as a final choice.

So that is why we chose Brightness 0.5 as our dataset color constancy so let's see the output from this method.

Example:

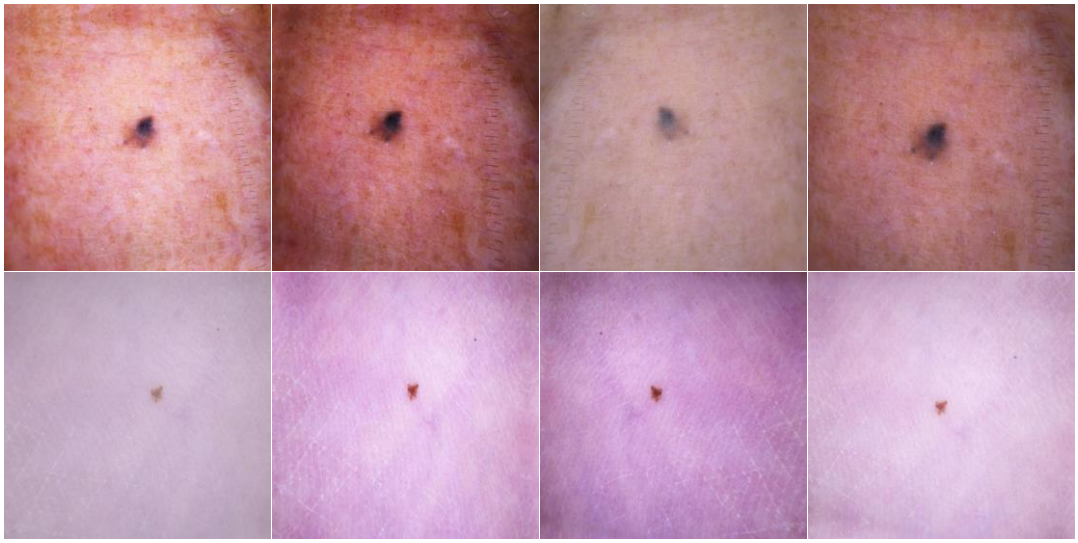
Real images:

Train1 ISIC_0027249 Test1 ISIC_0035417

Small Cancer example before brightness method (figure 50)

After methods:

Because it is random, we did it for the same image multiply times



Small cancer example after applying multiple times brightness method (figure 51)

Resize:

As far we deal with CNN models it is obvious that images must be resized to a size that allows CNN model to learn in acceptable time because deep learning taking time batch size and image size both control learning quality and time the most common size for most of the models is 224

Bigger images mean more computation operations per layer as well as more memory requirements so it depends on our resources and our CNN model size, we are already have limited resources in the tools that we chose, of course, we can handle this by batch size but this would take a long time and already our resources put a time limit of 12 hours so to learn our model we have to choose to mean size which is 224 the common one in learning deep learning models.

So, we can count this is the reason behind resize images because it far important in any deep learning to know the limits that we have and work with what we have

Example:

Real images:

Train1 ISIC_0027249 Test1 ISIC_0035417

Small cancer example before resizing (figure 52)

After method applied:

Train1 ISIC_0027249 Test1 ISIC_0035417

Small cancer example after resizing (figure 53)

Random Warp:

We simply chose some pre-processing and the random warp was in a lesson that we applied with the tool we put max warp to be 0.4, there is no reason more than it is just pre-processing for our dataset.

Image warping is the process of digitally manipulating an image such that any shapes portrayed in the image have been significantly distorted. Warping may be used for correcting image distortion as well as for creative purposes.

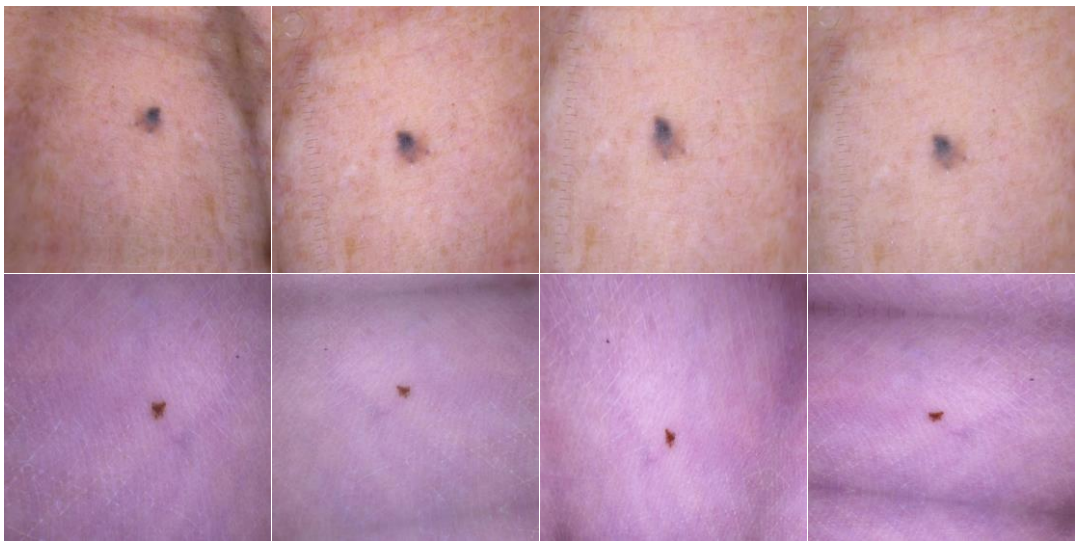
Example:

Real images:

Train1 ISIC_0027249 Test1 ISIC_0035417

Small cancer example before random warp (figure 54)

After method applied:



Small cancer example after applying multiple random warp (figure 55)

Random Rotate:

We simply chose some pre-processing and random rotate was in a lesson that we applied with the tool we put max warp to be 20 degrees, there is no reason more than it is just pre-processing for our dataset.

Image rotation is a common image processing routine with applications in matching, alignment, and other image-based algorithms. The input to an image rotation routine is an image, the rotation angle θ, and a point about which rotation is done [38] and because we have max degrees it will be like this: (max_rotate=angle) toggles random rotations between - angle to + angle specified here.

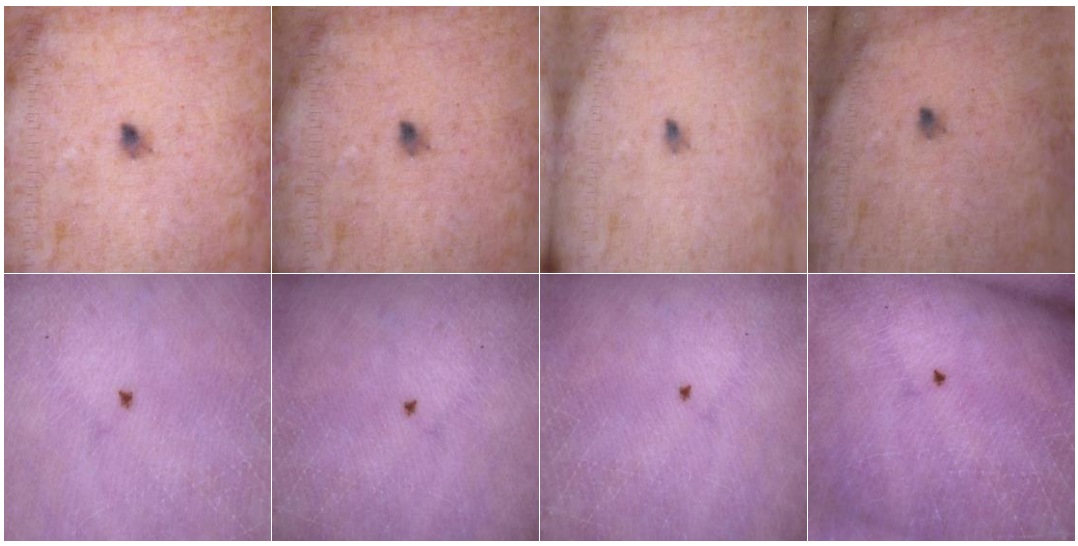
Example:

Real images:

Train1 ISIC_0027249 Test1 ISIC_0035417

Small cancer example before random rotate (figure 56)

After method applied:



Small cancer example after applying multiple random rotate (figure 57)

Conclusion:

Both versions one and three show some good advantages and small risks not like the second version which is very high risk and this will be proven later in the next chapter, as we proved that random brightness with max 0.5 was more promising than stretch and color equalization.

We also showed the importance of limited crop certain in resizing as well number of effected images by this method plus error given to method which will try to reduce error came from the segmentation model

Chapter 6 Training and test results

Introduction

We explained almost every reason for each choice and pre-processing architecture versions, in this chapter we will explain that we choose CNN as feature extractor and how we train our model and which approach we choose to do training and compare each output from each version, explain how to train approach effect or made this strange behavior that is in almost all models that we mentioned in the previous chapter as well as we will show results of each approaches with test time impact so we can explain strange behavior in both validation and test time as we are showing results of approaches in test.

Why CNN is better choice in our case?

As we explained in skin cancer Conclusion that cancer has many features and structure as well as an unknown classification that is near each other in each class that make sometimes confusion, we chose CNN [41] extractors as our feature extractor, because it will learn how to extract features that maximize the accuracy and minimize loss.

Why fastai as main tool in training?

We choose 1 cycle policy as the main approach in training due to our limitations it was more promising than other normal training approaches that were used in both TensorFlow and PyTorch.

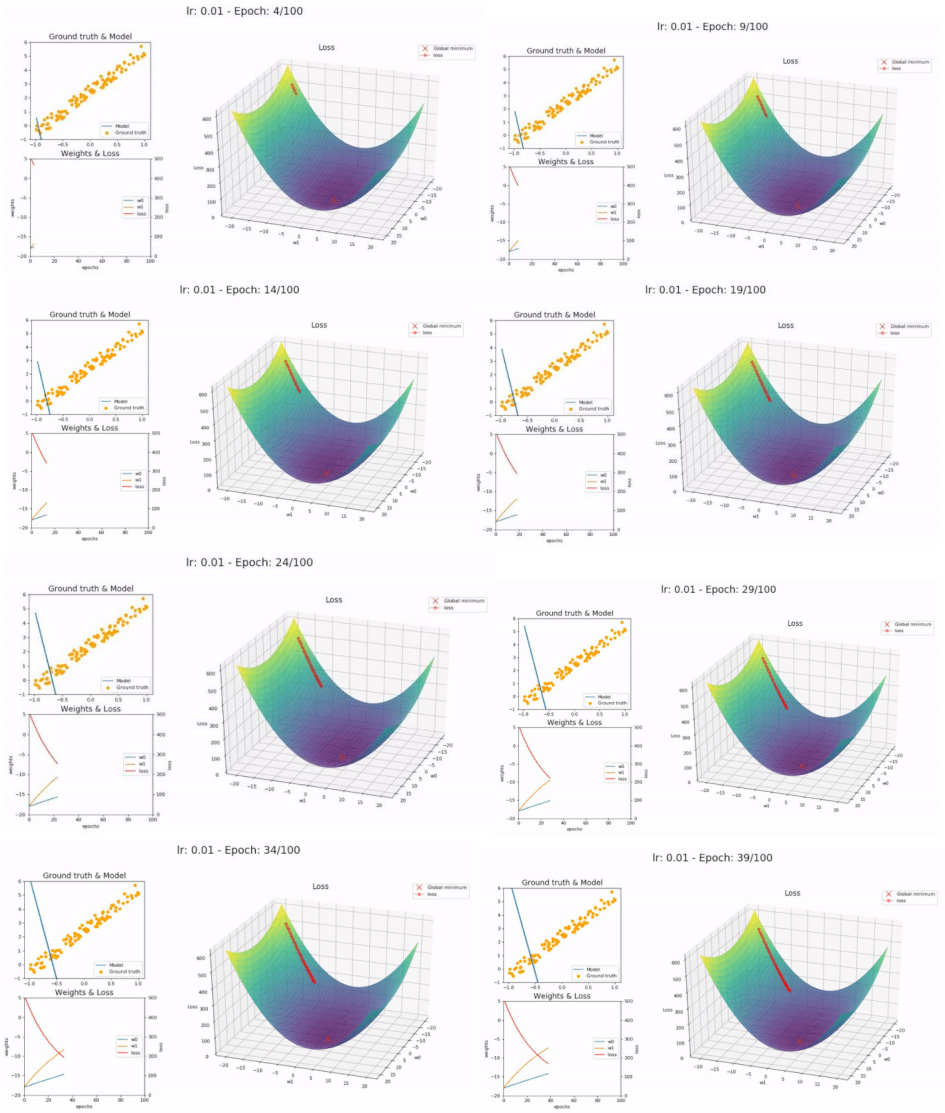
In the beginning we made baseline model in TensorFlow but we see poor results this why we go to PyTorch to have more controls in model fit function and build it from scratch and editing some parts yet our poor results improve by little improvement this when we try one fit cycle policy then we understand that this poor results was due to learning rate so if we stayed in PyTorch or TensorFlow we have to either find good learning rate or train more iterations which lead us to think about one cycle policy which is less in computation and less in training time before doing it ,we could only finish a model by luck due to google colab seasons that remain only 12 hours compared to fastai which implement 1 cycle policy it only takes from 4 to 5 hours to train single model and yet more effective on test time more than other approaches , we only chose fastai because it implement 1 cycle policy so after our try and error to realize that it took us long time to figure it out for this step more than planned to put.

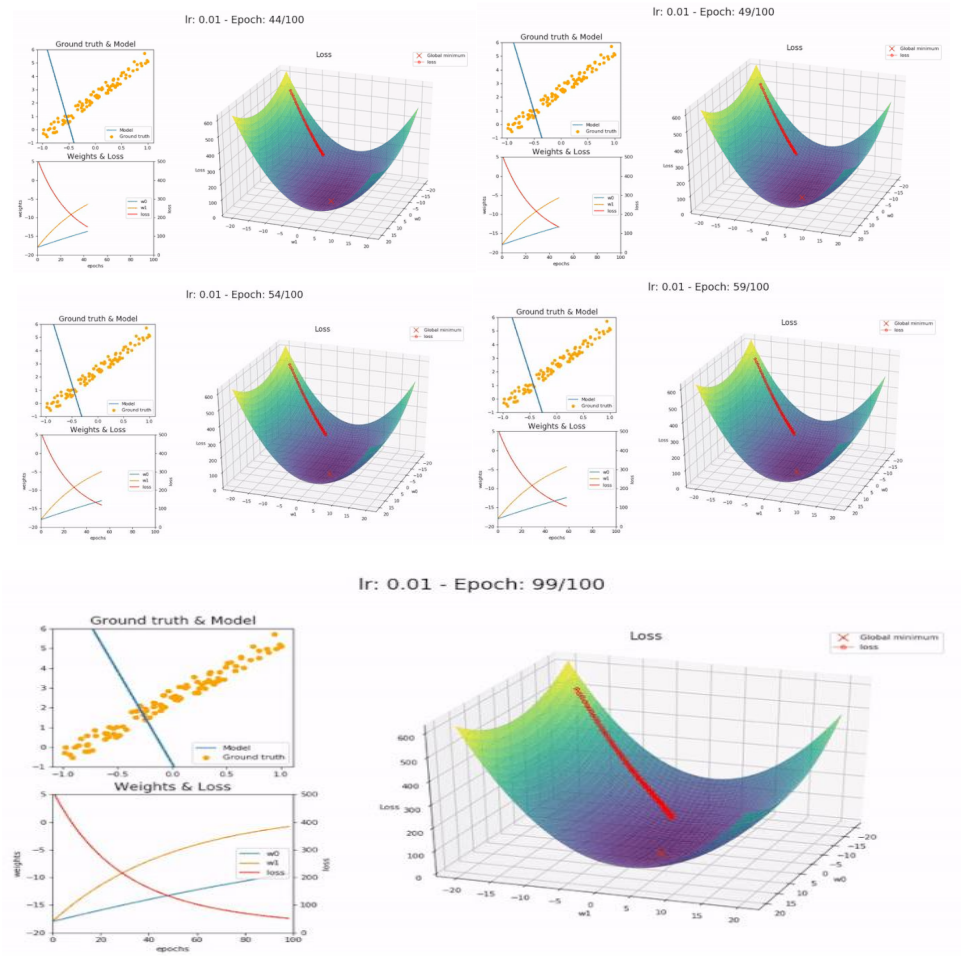
And to understand why one cycle policy is good in such cases to let's start by the problem of learning rate, cycling learning rates then Super-convergence.

The problem with Learning Rate

Training a Deep Neural Network (DNN) is a difficult global optimization problem. Learning Rate (LR) is a crucial hyper-parameter to tune when training DNNs. A very small learning rate can lead to very slow training, while a very large learning rate can hinder convergence as the loss function fluctuates around the minimum, or even diverges.

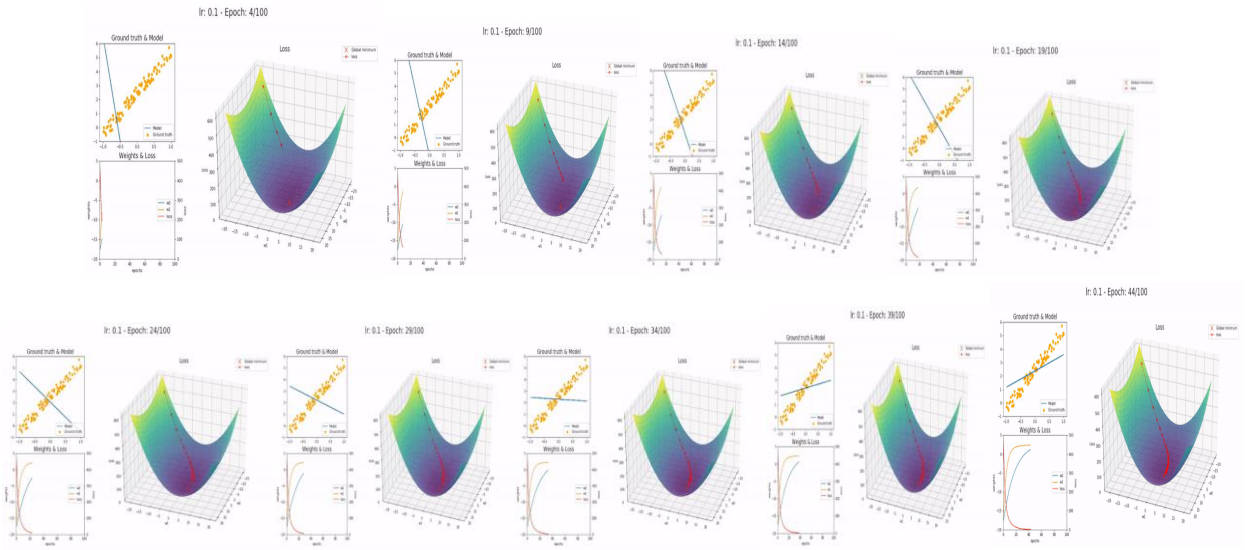
Too small LR (0.01). The model fails to converge within 100 epochs. More epochs—and time—required:

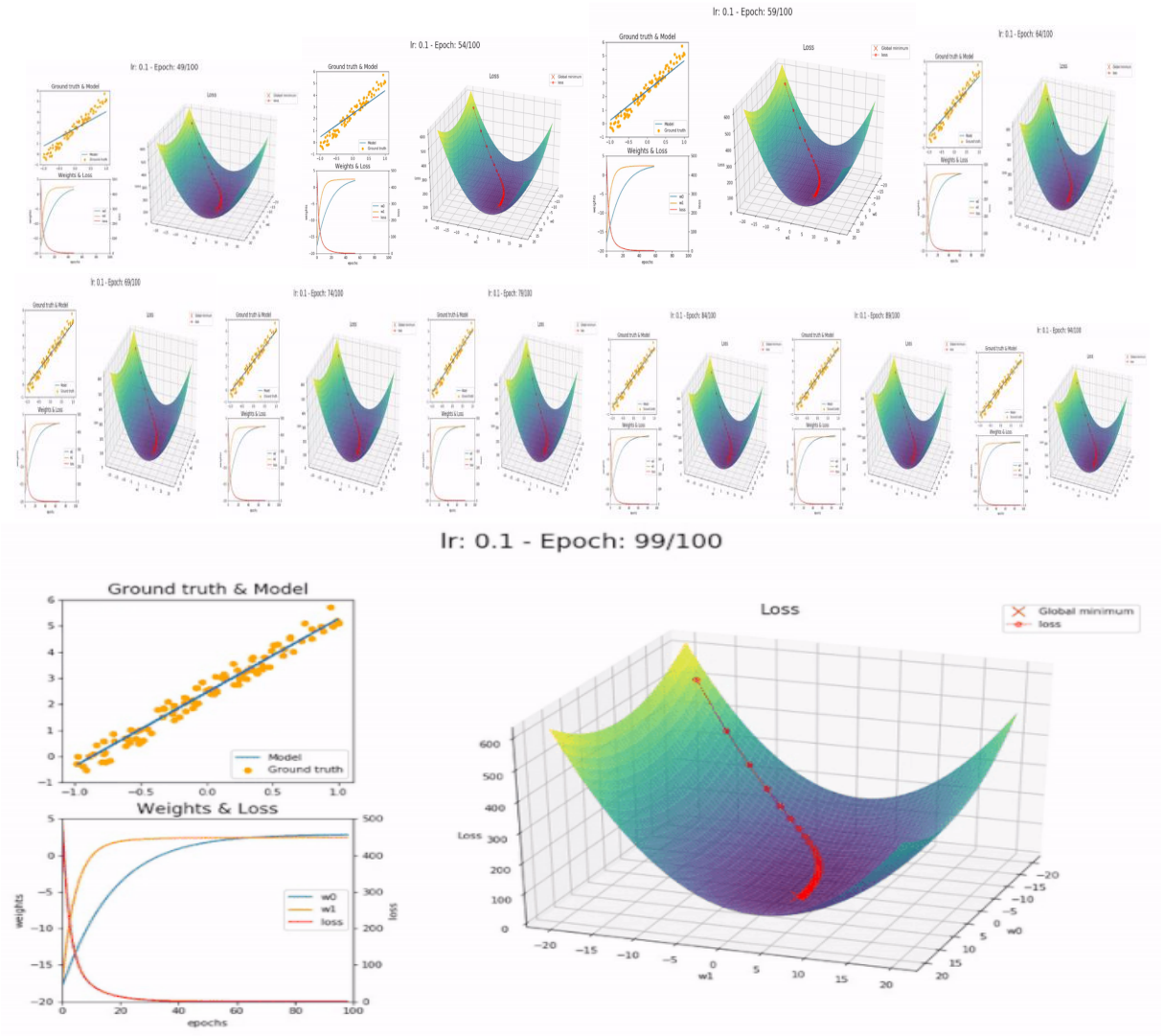




Model converge over epochs with small learning rate (figure 58)

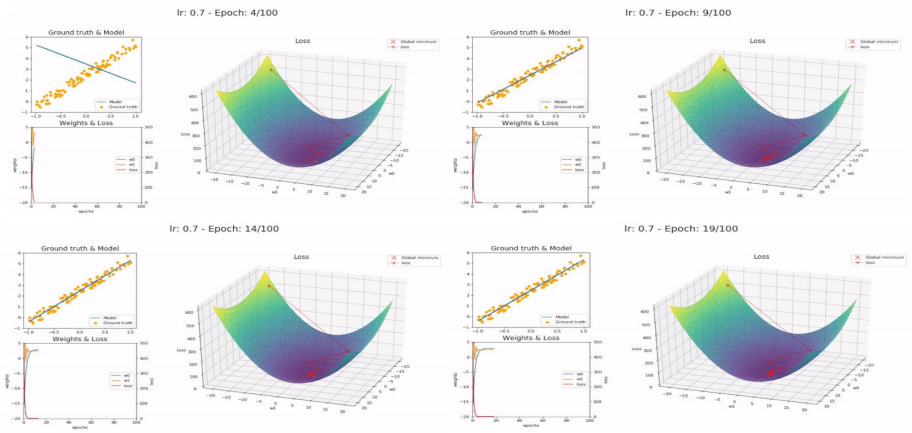
Good LR (0.1). The model converges successfully within 100 epochs as shown in figure:

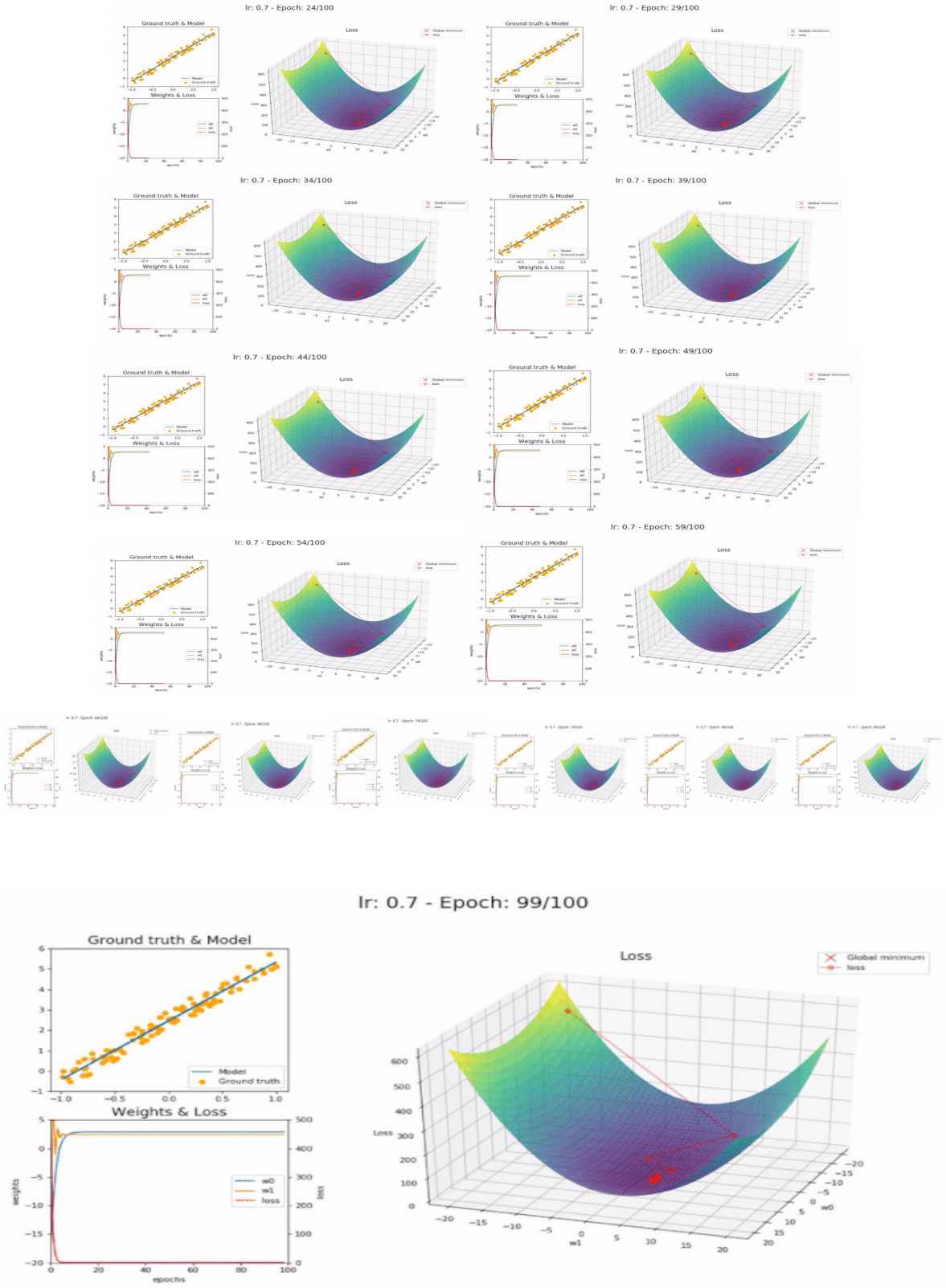




Model converge over epochs with good learning rate (figure 59)

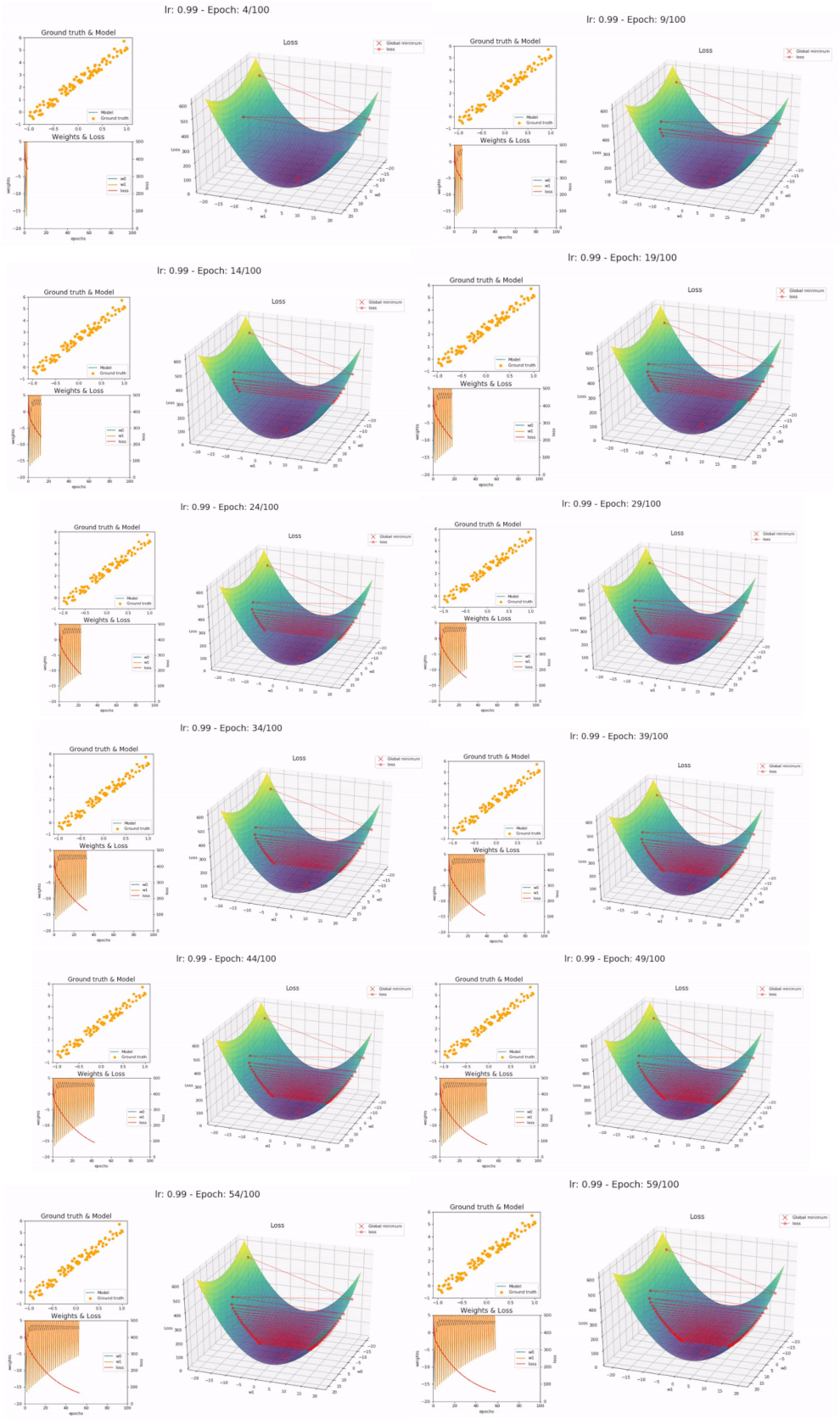
Optimal LR (0.7). The model converges successfully, very quickly, in under 10 epochs as shown in figure:

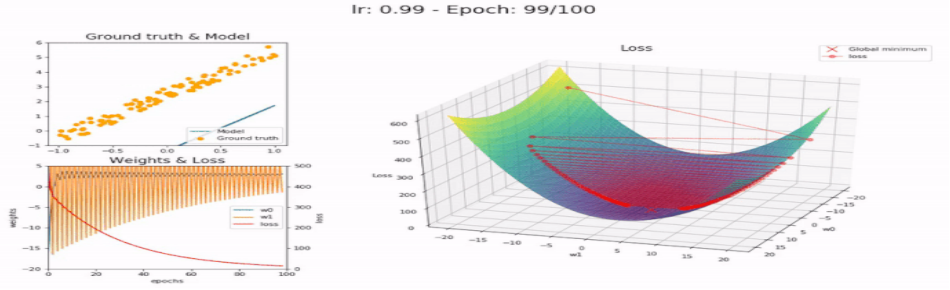




Model converge over epochs with optimal learning rate (figure 60)

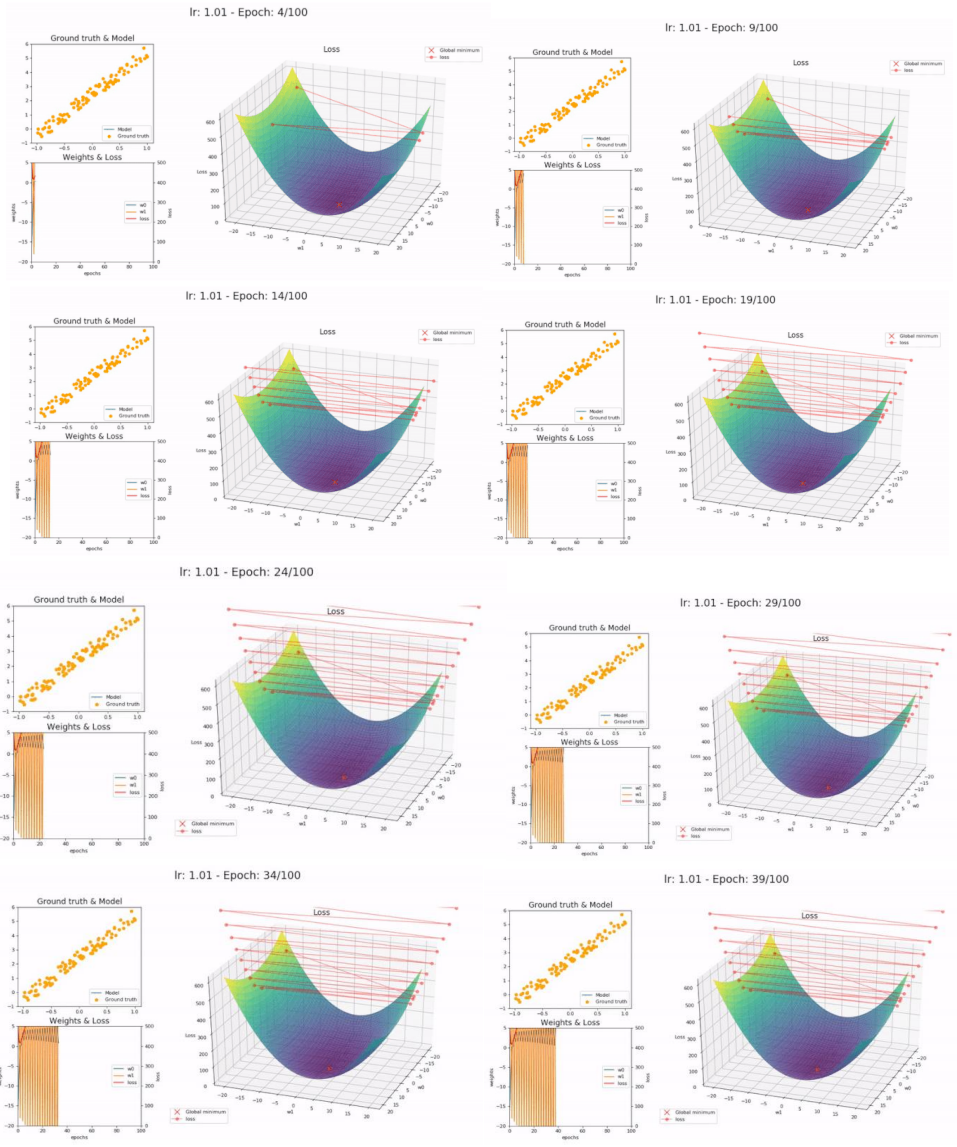
Large LR (0.99). The model fails to converge as the loss function fluctuates around the minimum as shown in figures:

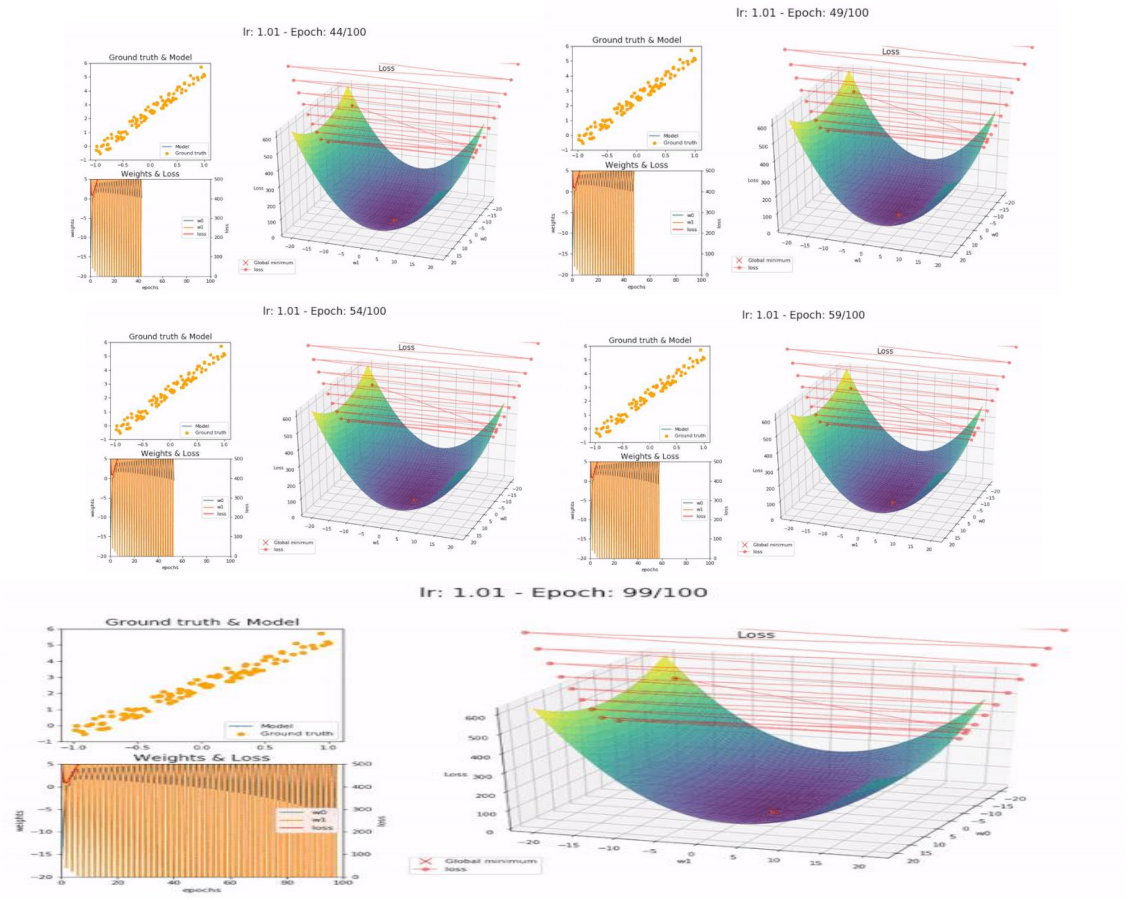




Model converge over epochs with large learning rate (figure 61)

Too large LR (1.01). The model diverges quickly as shown in figures:





Model converge over epochs with too large learning rate (figure 62)

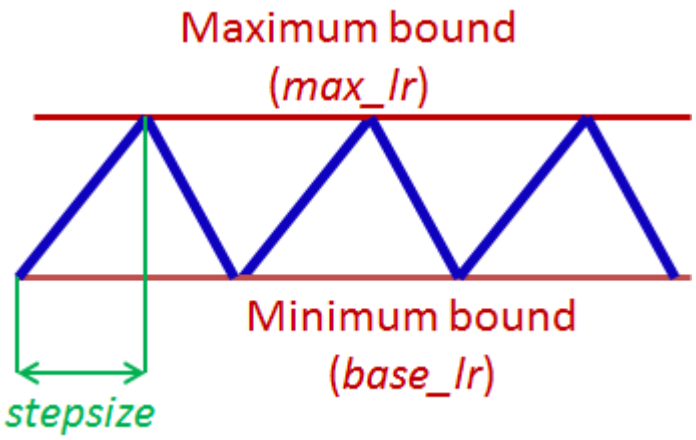
A low learning rate is slow but more accurate. As the learning rate increases so does the training speed, until the learning rate gets too large and diverges. Finding the sweet spot requires experimentation and patience. An automated way of calculating the optimal learning rate is to perform a grid search, but this is a time-consuming process.

In practice, the learning rate is not static but changes as training progress. It is desirable to start with an optimal learning rate (for speed) and gradually decrease it towards the end (for accuracy). There are two ways to achieve this: learning rate schedules and adaptive learning rate methods.

Learning rate schedules are mathematical formulas that decrease the learning rate using a particular strategy (Time-Based Decay, Step Decay, Exponential Decay, etc.). That strategy/schedule is set before training commences and remains constant throughout the training process. Thus, learning rate schedules are unable to adapt to the particular characteristics of a dataset. Adaptive learning rate methods (Adagrad, Adadelta, RMSprop, Adam, etc.) alleviate that problem but are computationally expensive.

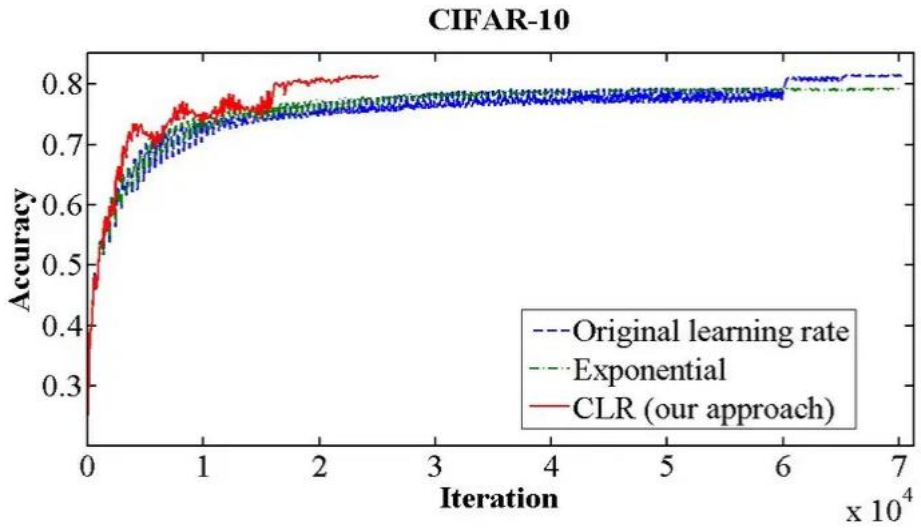
Cyclical Learning Rates

Smith discovered a new method for setting the learning rate, named Cyclical Learning Rates (CLRs). Instead of using a fixed, or a decreasing learning rate, the CLR method allows the learning rate to continuously oscillate between a reasonable minimum and maximum bound. One CLR cycle consists of two steps; one in which the learning rate increases and one in which it decreases. Each step has a size (called stepsize), which is the number of iterations (e.g. 1k, 5k, etc.) where the learning rate increases or decreases. Two steps form a cycle. Concretely, a CLR cycle with step size of 5,000 will consist of 5,000 + 5,000 = 10,000 total iterations. A CLR policy might consist of multiple cycles. [39]



CLR method (figure 63)

CLRs are not computationally expensive and eliminate the need to find the best learning rate value—the optimal learning rate will fall somewhere between the minimum and maximum bounds. A cyclical learning rate produces better overall results, despite the fact that it might hinder the network performance temporarily. [39]

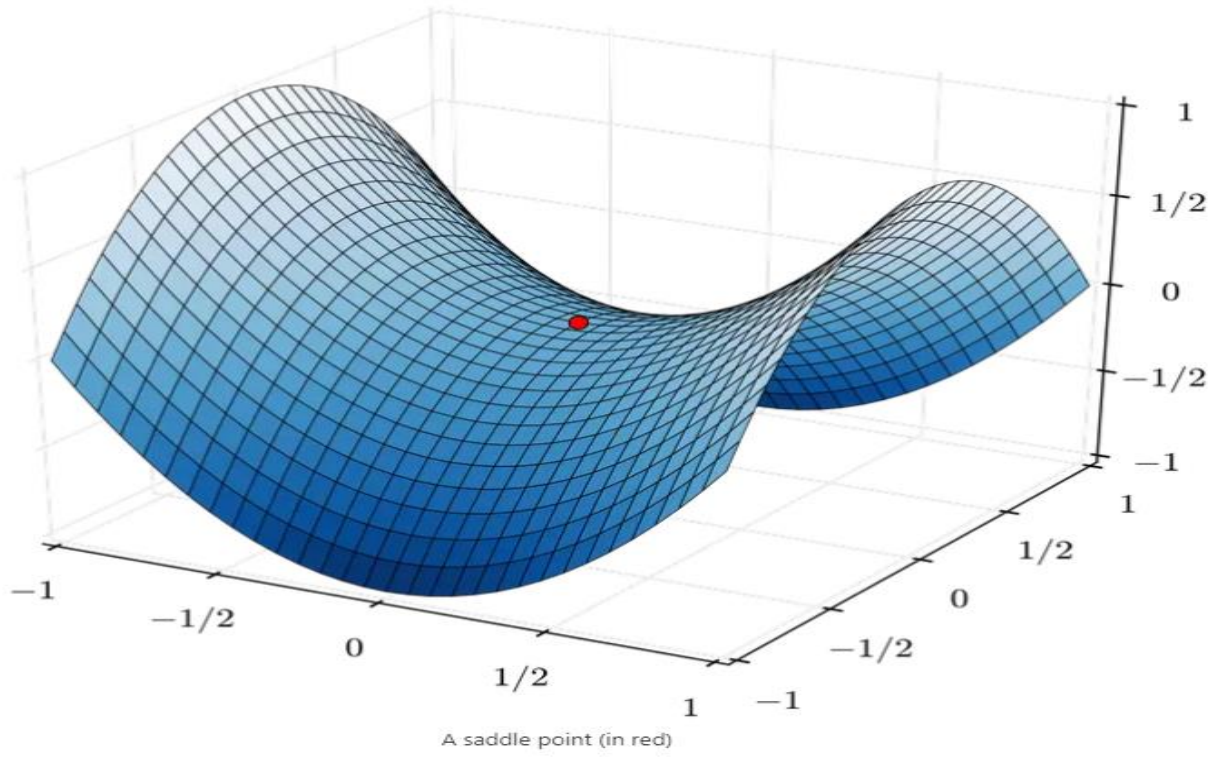


Accuracy over iteration on cifar-10 dataset (figure 64)

The above figure shows the training accuracy of the CIFAR-10 dataset over 70,000 iterations. A fixed learning rate (blue line) achieves 81.4% accuracy after 70,000 iterations, while the CLR method (red line) achieves the same within 25,000 iterations.[39]

“The essence of this learning rate policy comes from the observation that increasing the learning rate might have a short-term negative effect and yet achieve a longer-term beneficial effect.” [39]

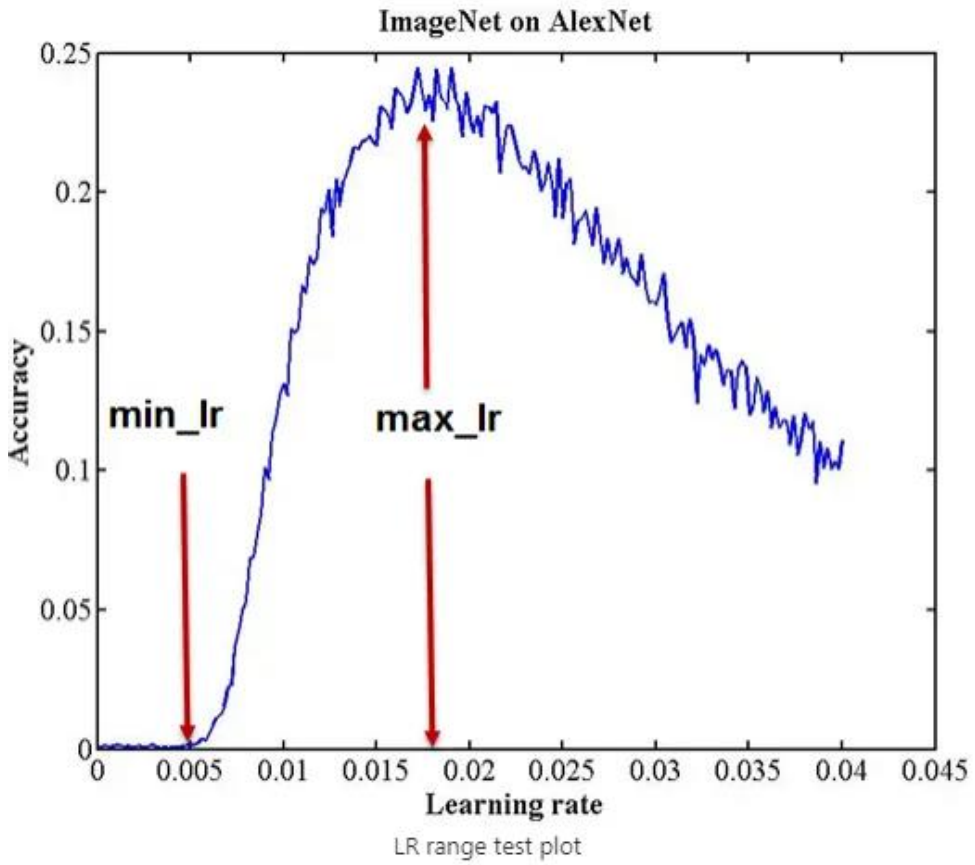
Cyclical Learning Rates are effective because they can successfully negotiate saddle points, which typically have small gradients (flat surfaces) and can slow down training when the learning rate is small. The best way to overcome such obstacles is to speed up and to move fast until a curved surface is found. The increasing learning rate of CLRs does just that, efficiently. [39]



Saddle point example (figure 65)

Learning Rate range test

Smith also devised a simple method for estimating reasonable minimum and maximum learning rate bounds; the LR range test. The test involves running a model for several epochs, where the learning rate starts at a low value and increases linearly towards a high value. A plot of accuracy versus learning rate shows when accuracy starts to increase and when it slows down, becomes ragged, or declines. The following LR range test plot shows two points that are good candidates for the minimum and maximum bounds [40]:



Accuracy over LR test (figure 66)

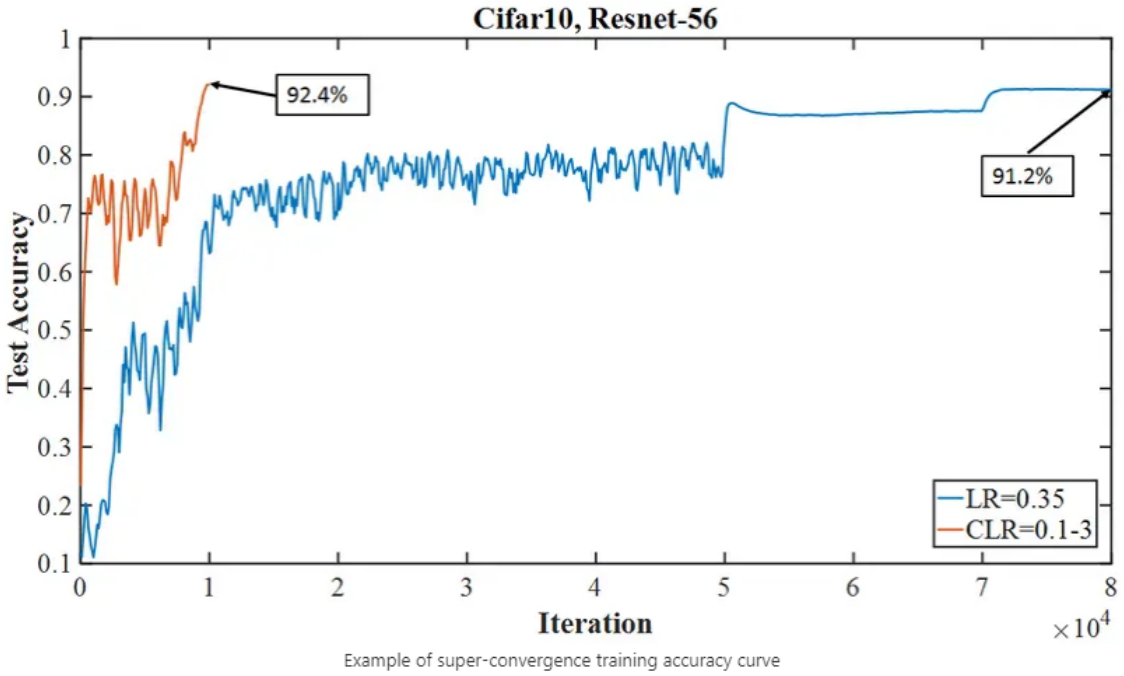
Subsequently, a Cyclical Learning Rate policy that varies between these bounds will produce good classification results, often with fewer iterations and without any significant computational expense, for a range of architectures.

Super-convergence and 1cycle policy

Building on his CLR research, Smith followed up with his paper on super-convergence [40], a phenomenon where neural networks can be trained an order of magnitude faster than with standard training methods [40].

Super-convergence uses the CLR method, but with just one cycle—which contains two learning rate steps, one increasing and one decreasing—and a large maximum learning rate bound. The cycle's size must be smaller than the total number of iterations/epochs. After the cycle is complete, the learning rate should decrease even further for the remaining iterations/epochs, several orders of magnitude less than its initial value. Smith named this the 1cycle policy [40].

Concretely, in super-convergence, the learning rate starts at a low value, increases to a very large value, and then decreases to a value much lower than its initial one [40]. The effect of that learning rate movement is a very distinctive training accuracy curve. Traditional training accuracy curves increase, then plateau as the value of learning rate changes (see the blue curve, below). Super-convergence training accuracy curves (see the red curve, below) have a dramatic initial jump (moving fast as learning rate increases), oscillate or even decline for a bit (while learning rate is very large) and then jump up again to a distinctive accuracy peak (as learning rate decreases to a very small value).



Example of super-convergence training accuracy curve

Test accuracy over iteration (figure 67)

Smith found that a large learning rate acts as a regularization method. Hence, when using the 1cycle policy, other regularization methods (batch size, momentum, weight decay, etc.) must be reduced. [40]

What hyper parameters that we can set in fastai and how to find it?

We just try and error number of epochs that maximize the test results

We set 3 stages to find the best number of epochs at learning of models

Stage 1 models will train on 33 epochs, train all models

Stage 2 models will train on 44 epochs, train all models

Stage 3 models will train on 55 epochs, train all models that show improvement in stage 2 more than 1 by 1 percentage at least

What is our loss function and reason behind?

We tried both weighted cross-entropy and focal loss function, in the beginning, we followed first place with public data competitor document that recommends that focal loss worth to try so we tried it but the space in test impact and validation time impact was too big for weighted cross entropy more than focal loss .but we missed hyper parameters of focal loss we just tried only one case but we think that optimal parameters will do as or more than weighted cross entropy but it would take too much try and error so we did not take the risk because we have already weighted cross-entropy as our main loss, and our research is not talking about how to tuning existing methods but creating new solution methods

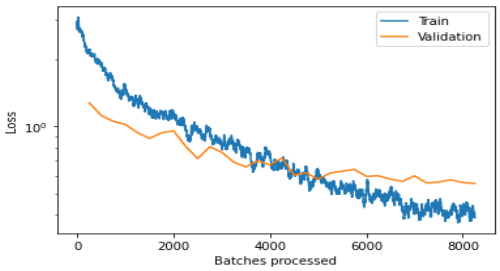
Results of each learning stages:

33 epochs:

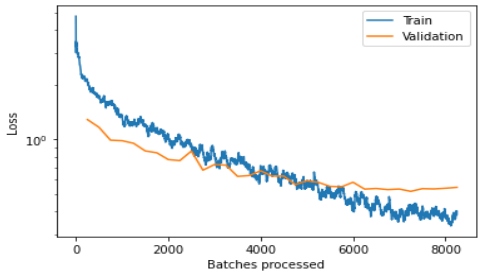
Pre-processing Version 1(limited crop certain) models:

Densnet:

Densnet169 Validation -> 82 mean recall (macro)



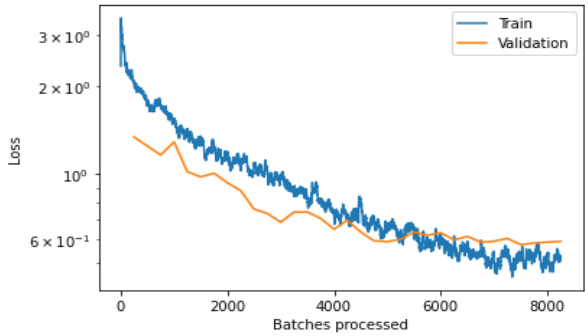
Densnet201 Validation -> 82 mean recall (macro)



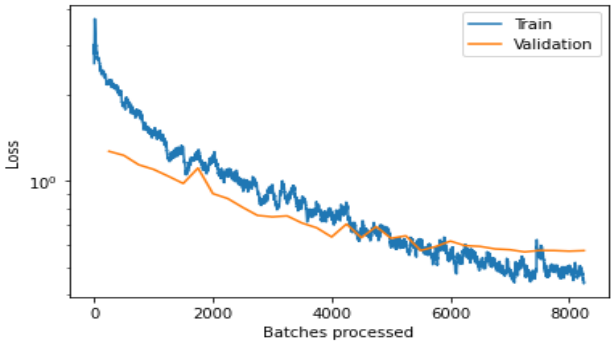
Loss over batches processed Densnet v1model33epoch (figure 68)

Resent:

Resnet101 Validation -> 78 mean recall (macro)



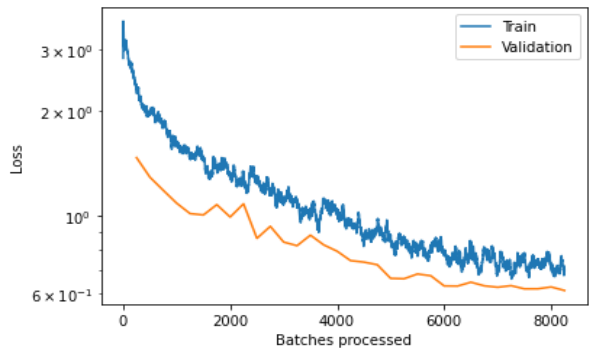
Resnet152 Validation -> 80 mean recall (macro)



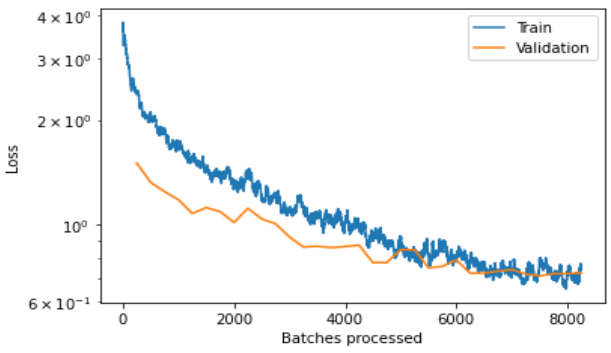
Loss over batches processed Resnet v1model33epoch (figure 69)

Vgg:

Vgg16 Validation -> 77 mean recall (macro)



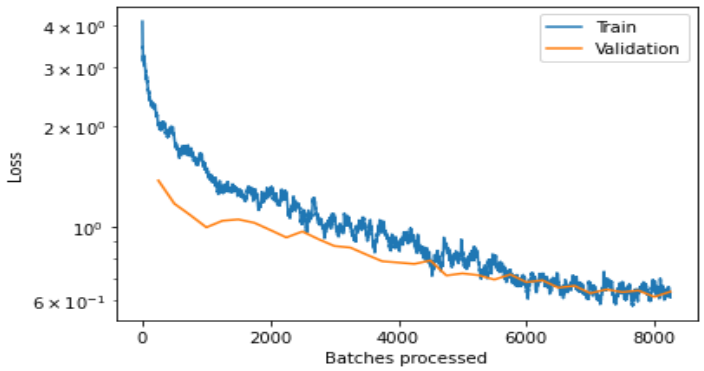
Vgg19 Validation -> 76 mean recall (macro)



Loss over batches processed Vgg v1model33epoch (figure 70)

Senet:

Senet154 Validation -> 77 mean recall (macro)

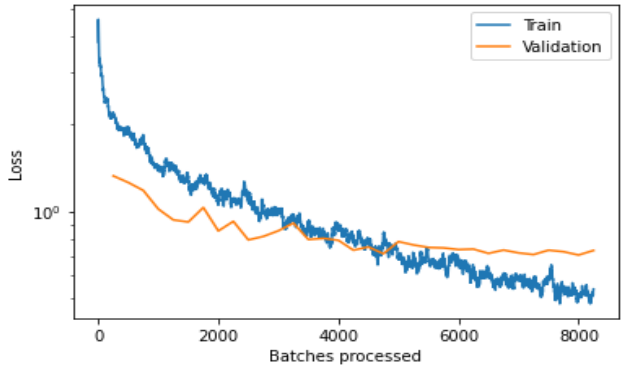


Loss over batches processed Senet v1model33epoch (figure 71)

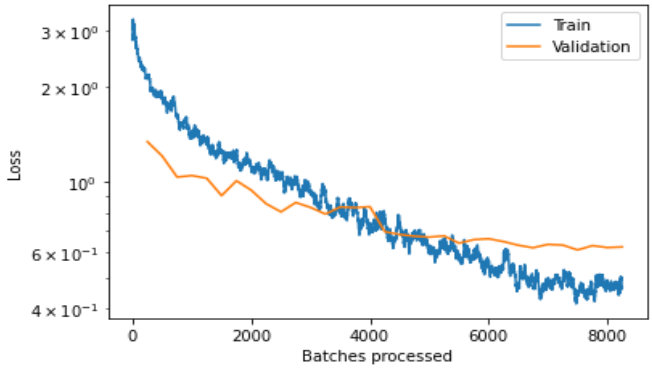
Pre-processing Version 2(segmentation mask bitwise) models:

Densnet:

Densnet169 Validation -> 76 mean recall (macro)



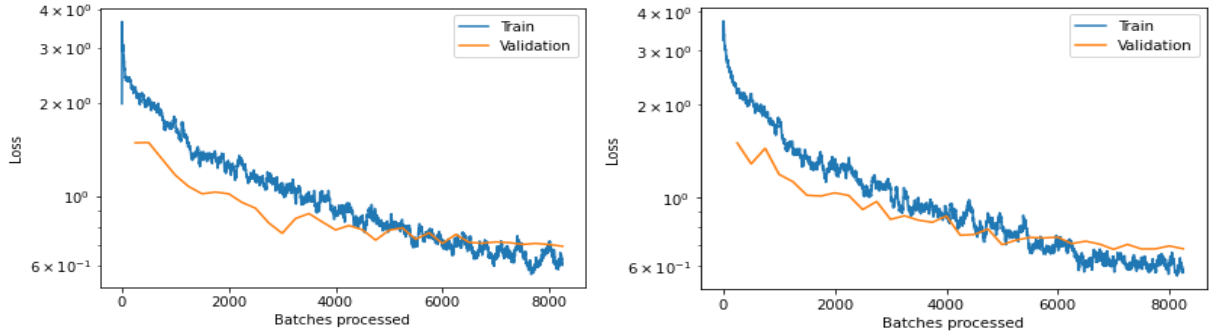
Densnet201 Validation -> 81 mean recall (macro)



Loss over batches processed Densnet v2model33epoch (figure 72)

Resent:

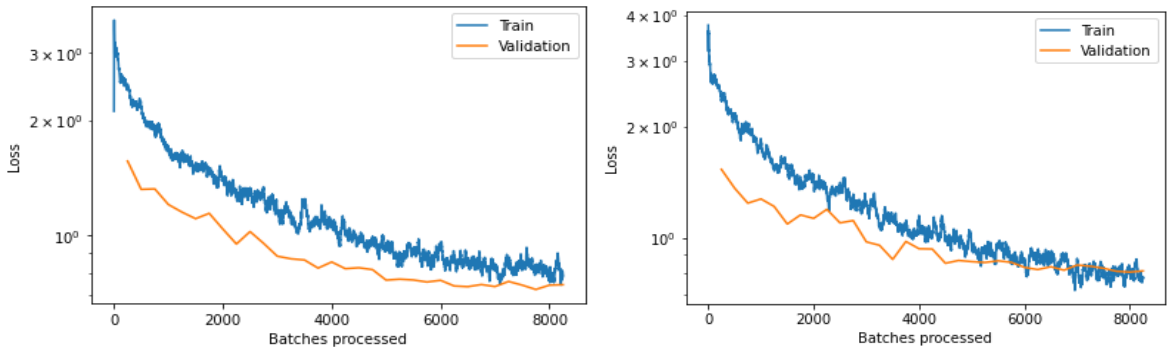
Resnet101 Validation -> 72 mean recall (macro) **Resnet152 Validation -> 75 mean recall (macro)**



Loss over batches processed Resnet v2model33epoch (figure 73)

Vgg:

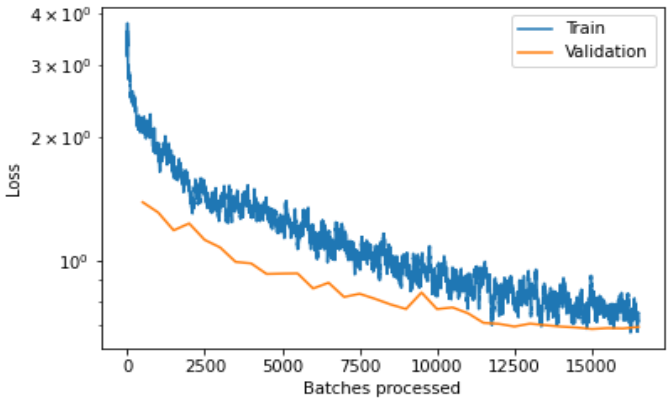
Vgg16 Validation -> 72 mean recall (macro) **Vgg19 Validation -> 70 mean recall (macro)**



Loss over batches processed Vgg v2model33epoch (figure 74)

Senet:

Senet154 Validation -> 73 mean recall (macro)

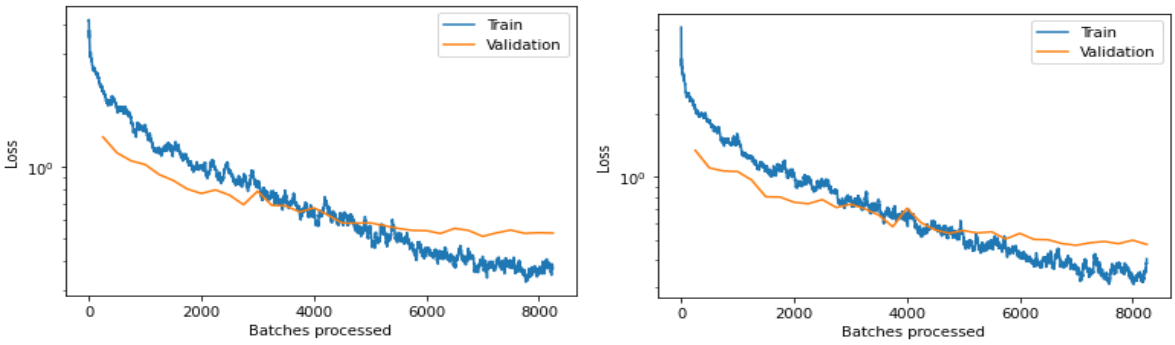


Loss over batches processed Senet v2model33epoch (figure 75)

Pre-processing Version 3(zoom) models:

Densnet:

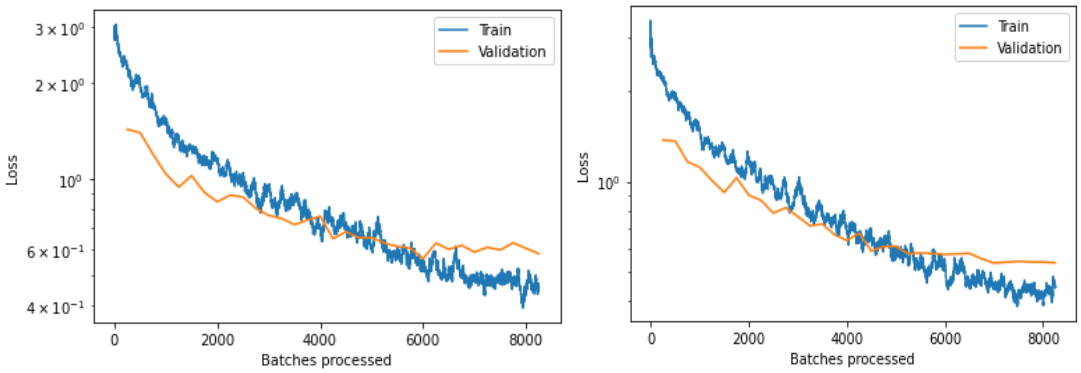
Densnet169 Validation ->81 mean recall (macro) **Densnet201 Validation ->83 mean recall (macro)**



Loss over batches processed Densnet v3model33epoch (figure 76)

Resent:

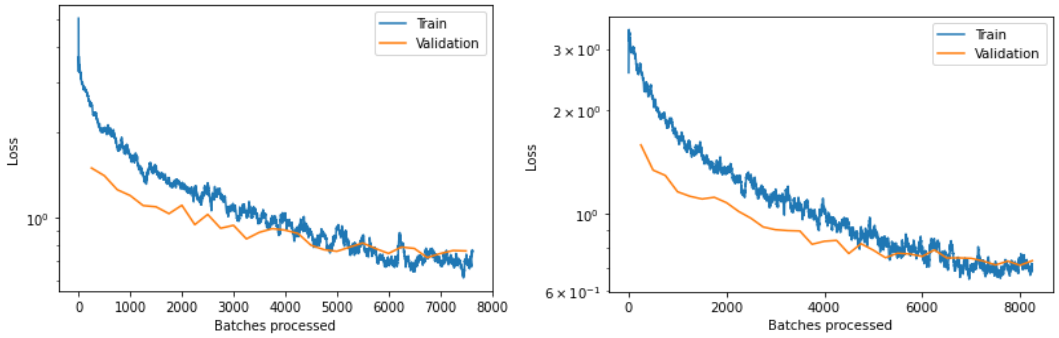
Resnet101 Validation -> 79 mean recall (macro) **Resnet152 Validation -> 82 mean recall (macro)**



Loss over batches processed Resnet v3model33epoch (figure 77)

Vgg:

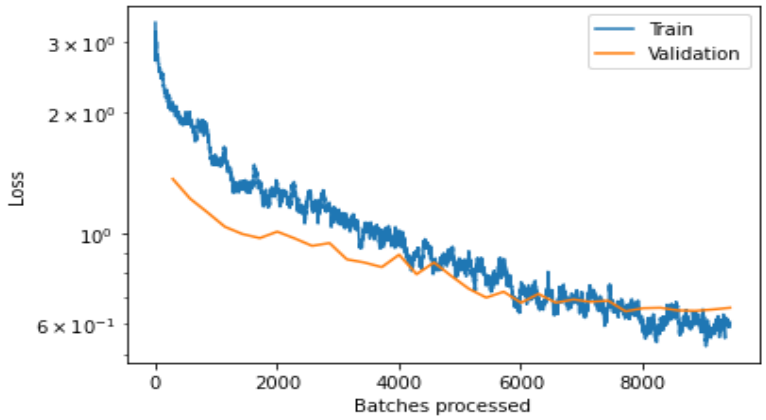
Vgg16 Validation -> 77 mean recall (macro) **Vgg19 Validation -> 72 mean recall (macro)**



Loss over batches processed Vgg v3model33epoch (figure 78)

Senet:

Senet154 Validation -> 76 mean recall (macro)



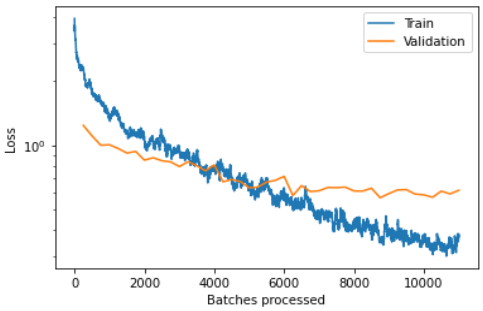
Loss over batches processed Senet v3model33epoch (figure 79)

44 epochs:

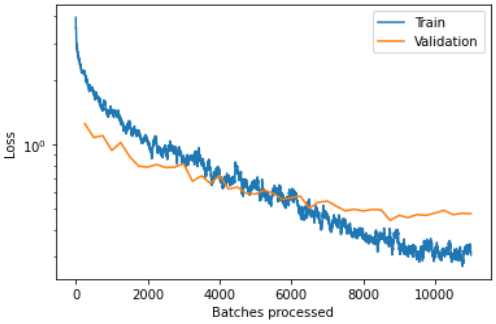
Pre-processing Version 1(limited crop certain) models:

Densnet:

Densnet169 Validation -> 80 mean recall (macro)



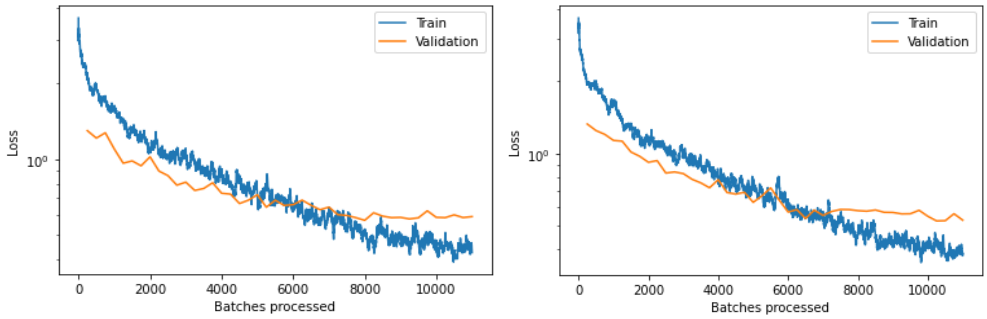
Densnet201 Validation -> 83 mean recall (macro)



Loss over batches processed Densnet v1model44epoch (figure 80)

Resent:

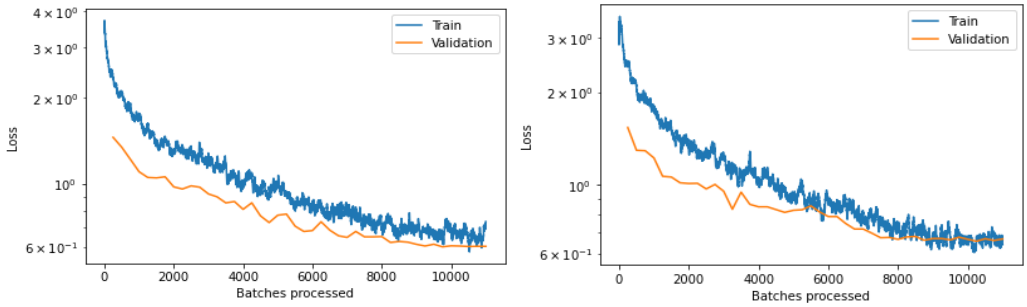
Resnet101 Validation -> 79 mean recall (macro) **Resnet152 Validation -> 83 mean recall (macro)**



Loss over batches processed Resnet v1model44epoch (figure 81)

Vgg:

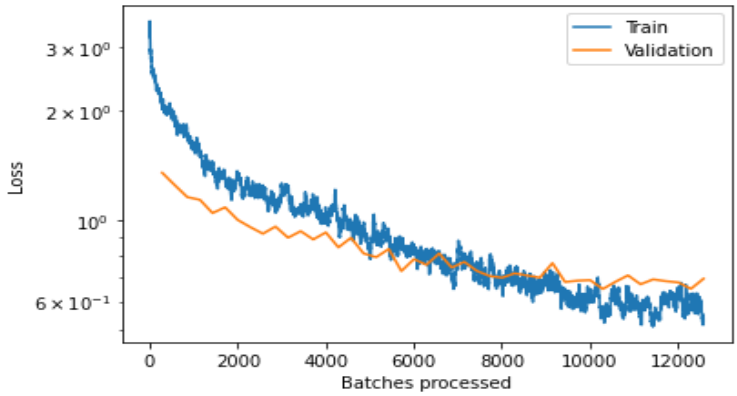
Vgg16 Validation -> 78 mean recall (macro) **Vgg19 Validation -> 77 mean recall (macro)**



loss over batches processed Vgg v1model44epoch (figure 82)

Senet:

Senet154 Validation -> 77 mean recall (macro)

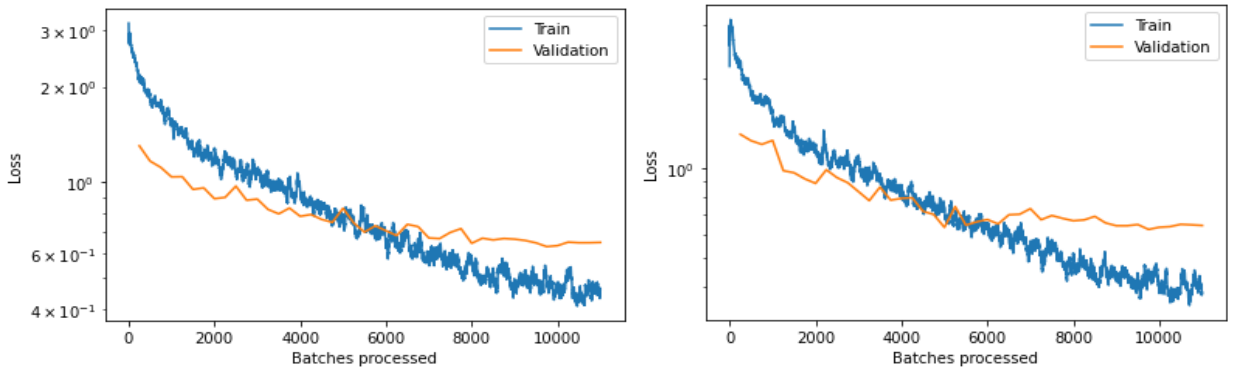


loss over batches processed Senet v1model44epoch (figure 83)

Pre-processing Version 2(segmentation mask bitwise) models

Densnet:

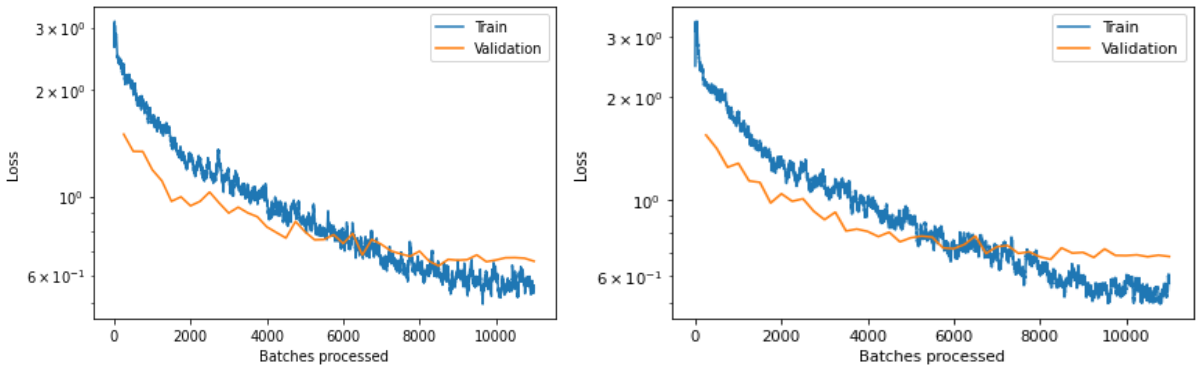
Densnet169 Validation -> 77 mean recall (macro) **Densnet201 Validation -> 78 mean recall (macro)**



loss over batches processed Densnet v2model44epoch (figure 84)

Resent:

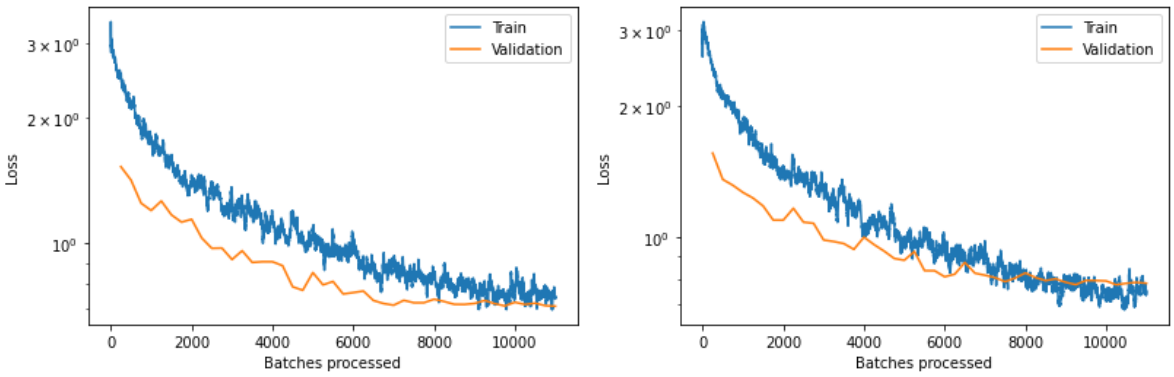
Resnet101 Validation -> 74 mean recall (macro) **Resnet152 Validation -> 76 mean recall (macro)**



loss over batches processed Resnet v2model44epoch (figure 85)

Vgg:

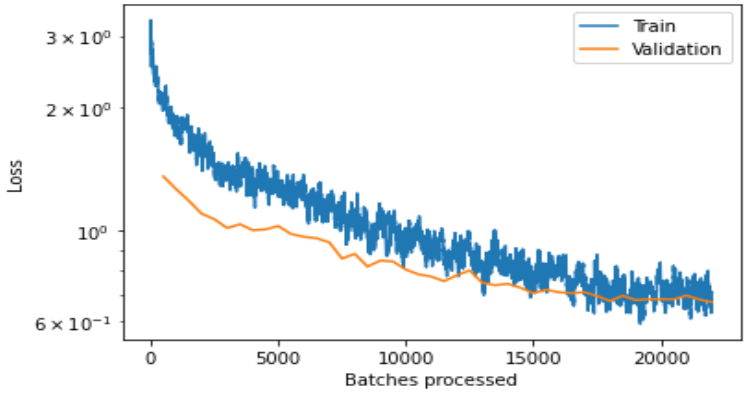
Vgg16 Validation -> 73 mean recall (macro) **Vgg19 Validation -> 71 mean recall (macro)**



loss over batches processed Vgg v2model44epoch (figure 86)

Senet:

Senet154 Validation -> 75 mean recall (macro)

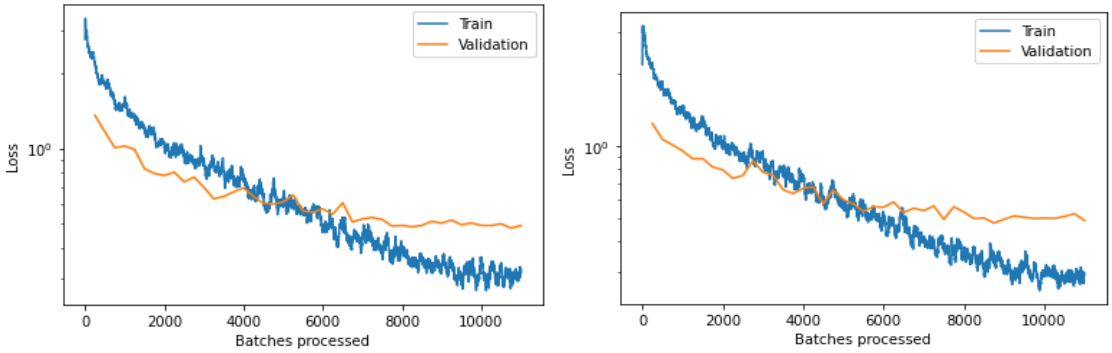


loss over batches processed Senet v2model44epoch (figure 87)

Pre-processing Version 3(zoom) models:

Densnet:

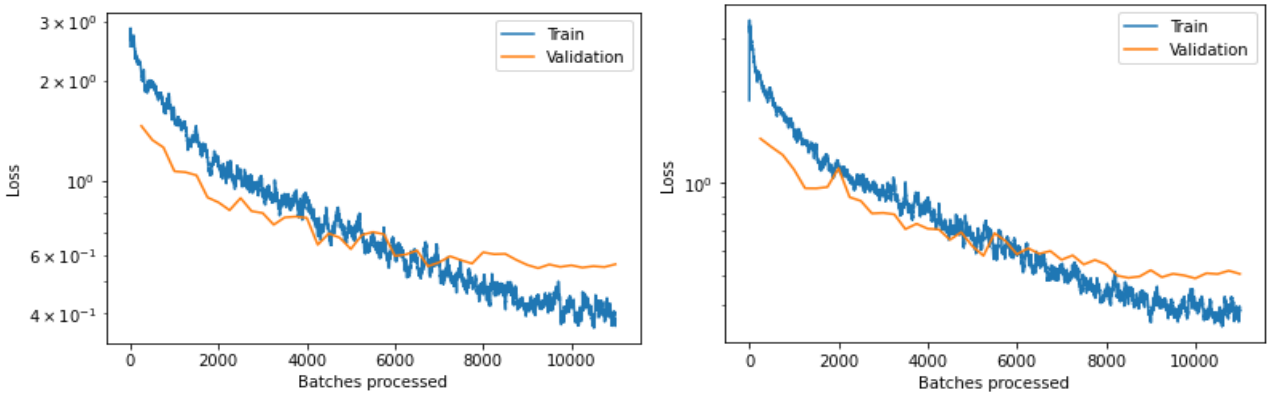
Densnet169 Validation ->83 mean recall (macro) **Densnet201 Validation ->83 mean recall (macro)**



loss over batches processed Densnet v3model44epoch (figure 88)

Resent:

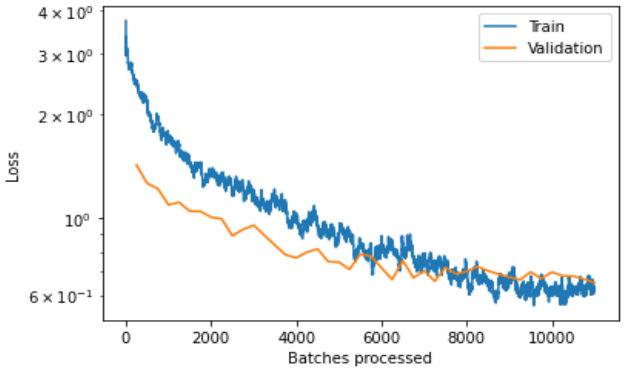
Resnet101 Validation -> 80 mean recall (macro) **Resnet152 Validation -> 82 mean recall (macro)**



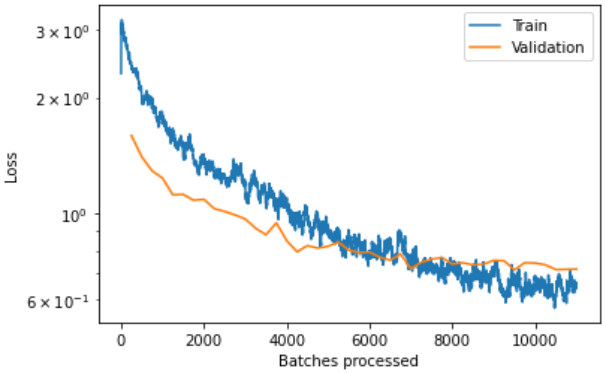
loss over batches processed Resnet v3model44epoch (figure 89)

Vgg:

Vgg16 Validation -> 77 mean recall (macro)



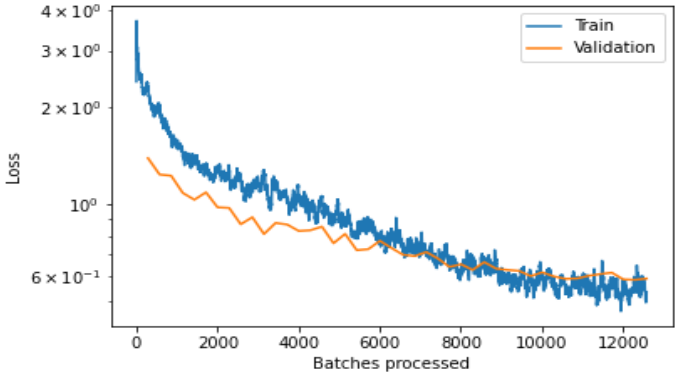
Vgg19 Validation -> 74 mean recall (macro)



loss over batches processed Vgg v3model44epoch (figure 90)

Senet:

Senet154 Validation -> 78 mean recall (macro)



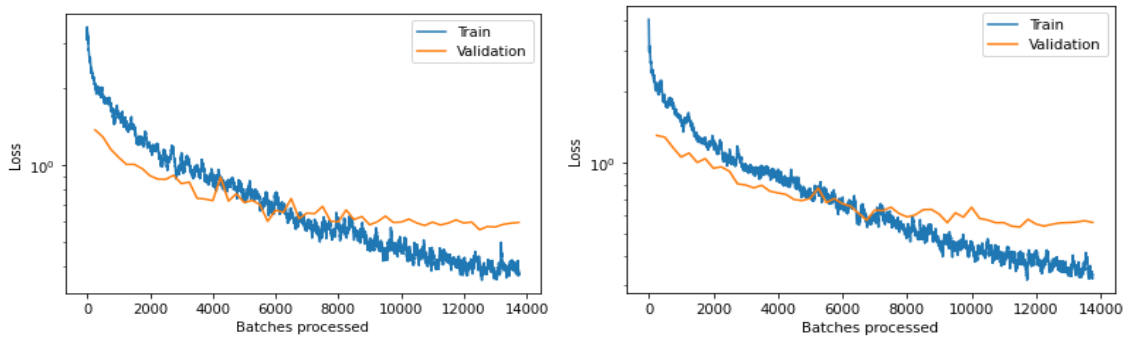
loss over batches processed Senet v3model44epoch (figure 91)

55 epochs:

Pre-processing Version 1(limited crop certain) models

Resnet:

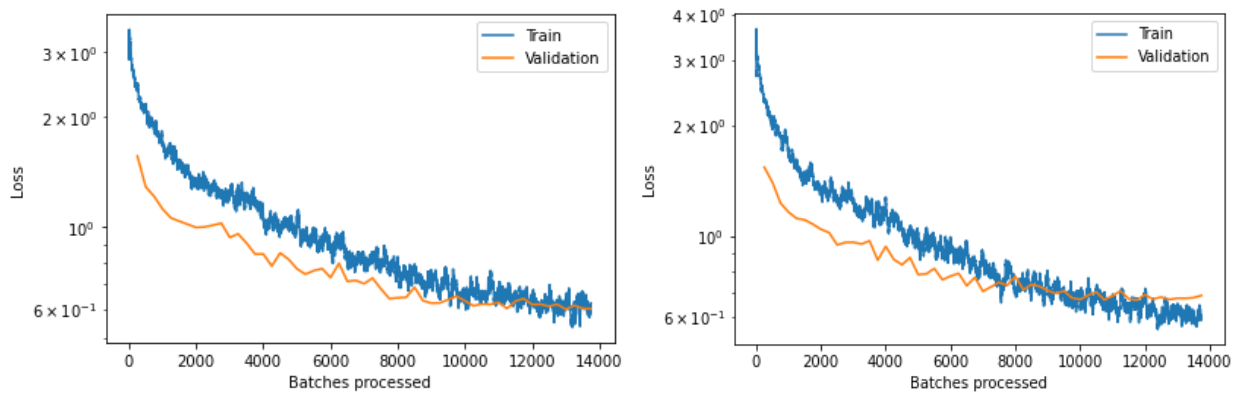
Resnet101 Validation -> 81 mean recall (macro) **Resnet152 Validation ->80 mean recall (macro)**



loss over batches processed Resnet v1model55epoch (figure 92)

Vgg:

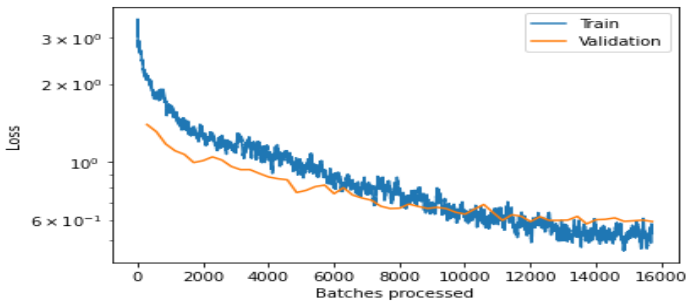
Vgg16 Validation -> 80 mean recall (macro) **Vgg19 Validation -> 78 mean recall (macro)**



loss over batches processed Vgg v1model55epoch (figure 93)

Senet:

Senet154 Validation -> 81 mean recall (macro)

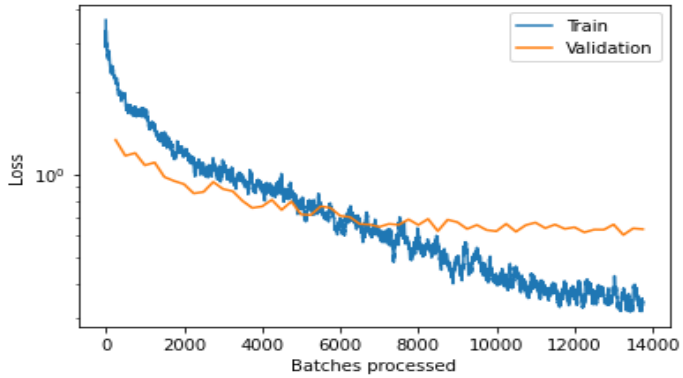


loss over batches processed Senet v1model55epoch (figure 94)

Pre-processing Version 2(segmentation mask bitwise) models:

Densnet:

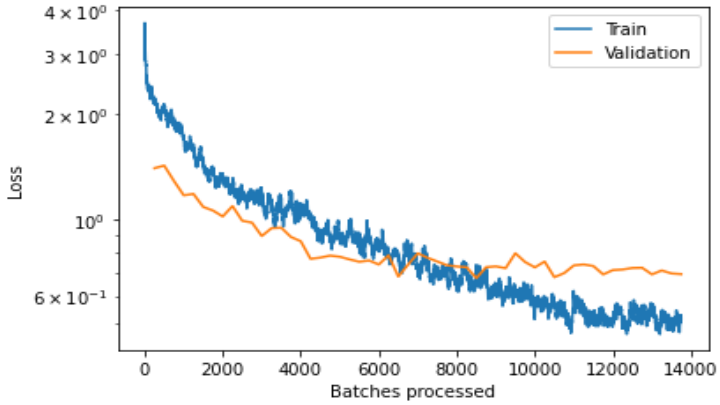
Densnet201 Validation -> 78 mean recall (macro)



loss over batches processed Densnet v2model55epoch (figure 95)

Resnet:

Resnet101 Validation -> 72 mean recall (macro)

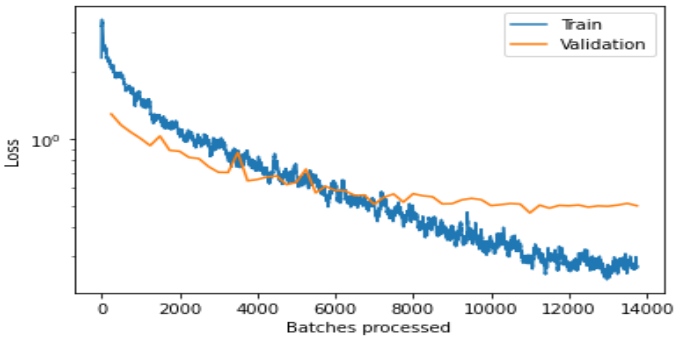


loss over batches processed Resnet v2model55epoch (figure 96)

Pre-processing Version 3(zoom) models:

Densnet:

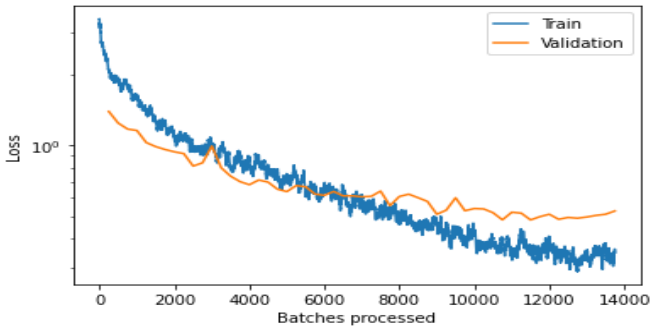
Densnet169 Validation ->84 mean recall (macro)



loss over batches processed Densnet v3model55epoch (figure 97)

Resnet:

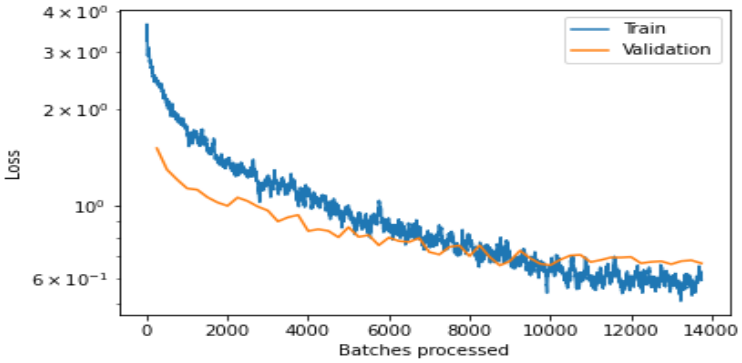
Resnet152 Validation ->80 mean recall (macro)



loss over batches processed Resnet v3model55epoch (figure 98)

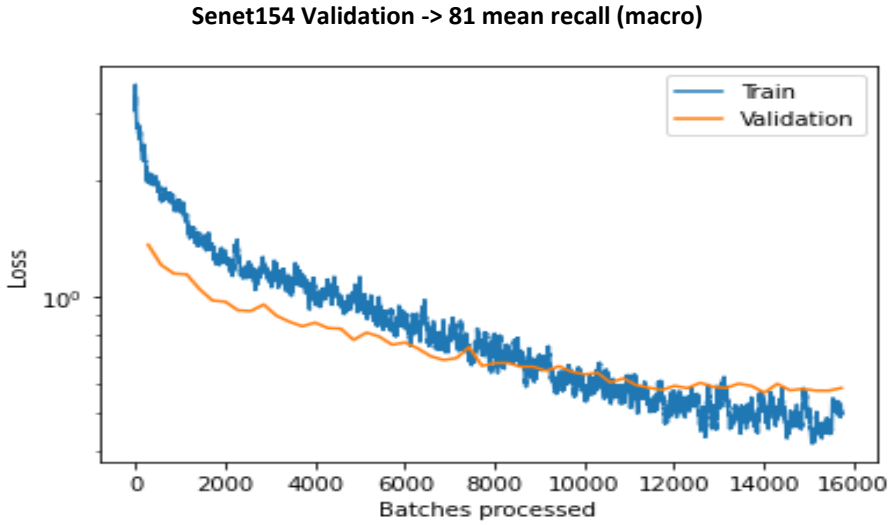
Vgg:

Vgg16 Validation -> 76 mean recall (macro)



loss over batches processed Vgg v3model55epoch (figure 99)

Senet:



Loss over batches processed Senet v3model55epoch (figure 100)

We could see each model loss improvement to batches processed and validation score now we will combine all this in small tables so we could make our conclusion.

Tables of validation and best test impact

We take all validation results and put it in tables, so each stage of 33 epochs, 44 epochs, and 55 epochs in each approaches to validation impact in each approach in our pre-processing approaches and we added which one was the best in test impact so we can discuss something that when we tried to train on our metadata using genetic algorithm as optimizer and validation predication as a fitness function in genetic algorithm. As shown in previous models figures and in the next figures, we put each approach by its unique pre-processing name data as the name of each table with validation.

limited crop certain validation table

epochs / model	33 epochs	44 epochs	55 epochs	**best test impact**
Densnet169	0.82	0.80	none	33 epochs
Densnet201	0.82	0.83	none	33 epochs
Resnet101	0.78	0.79	0.81	44 epochs
Resnet152	0.80	0.83	0.80	44 epochs
Vgg16	0.77	0.78	0.80	55 epochs
Vgg19	0.76	0.77	0.78	44 epochs
Senet154	0.77	0.77	0.81	55 epochs

bitwise validation table

epochs / model	33 epochs	44 epochs	55 epochs	**best test impact**
Densnet169	0.76	0.77	none	33 epochs
Densnet201	0.81	0.78	0.78	44 epochs
Resnet101	0.72	0.74	0.72	44 epochs
Resnet152	0.75	0.76	none	44 epochs
Vgg16	0.72	0.73	none	44 epochs
Vgg19	0.70	0.71	none	33 epochs
Senet154	0.73	0.75	none	44 epochs

zoom validation table

epochs / model	33 epochs	44 epochs	55 epochs	**best test impact**
Densnet169	0.81	0.83	0.84	55 epochs
Densnet201	0.83	0.84	none	33 epochs
Resnet101	0.79	0.80	none	33 epochs
Resnet152	0.82	0.82	0.82	44 epochs
Vgg16	0.77	0.77	0.76	44 epochs
Vgg19	0.72	0.74	none	33 epochs
Senet154	0.76	0.78	0.81	44 epochs

Limited crop certain/bitwise/zoom validation table (figure 101)

Form tables we could understand that almost the more epochs the more results but best test impact have another view that suggests that there is small overfitting in our models due to our train method or our method could have an effect that cannot be calculated in validation itself because one cycle policy reduces chances of over-fitting and also improves the model performance (model generalizes, instead of memorizing the data) due to using large learning rate in one cycle policy so a number of epochs affect the final result on the test so we need test results on each epoch of stages

limited crop certain Test table

epochs / model	33 epochs	44 epochs	55 epochs	**best epochs**
Densnet169	0.751	0.721	none	33 epochs
Densnet201	0.770	0.752	none	33 epochs
Resnet101	0.732	0.746	0.734	44 epochs
Resnet152	0.738	0.769	0.758	44 epochs
Vgg16	0.709	0.728	0.744	55 epochs
Vgg19	0.710	0.723	0.716	44 epochs
Senet154	0.700	0.715	0.734	55 epochs

bitwise Test table

epochs / model	33 epochs	44 epochs	55 epochs	**best epochs**
Densnet169	0.719	0.713	none	33 epochs
Densnet201	0.717	0.742	0.725	44 epochs
Resnet101	0.707	0.731	0.712	44 epochs
Resnet152	0.711	0.712	none	44 epochs
Vgg16	0.710	0.716	none	44 epochs
Vgg19	0.695	0.694	none	33 epochs
Senet154	0.709	0.713	none	44 epochs

zoom Test table

epochs / model	33 epochs	44 epochs	55 epochs	**best epochs**
Densnet169	0.760	0.765	0.779	55 epochs
Densnet201	0.776	0.757	none	33 epochs
Resnet101	0.756	0.730	none	33 epochs
Resnet152	0.761	0.770	0.754	44 epochs
Vgg16	0.716	0.741	0.724	44 epochs
Vgg19	0.691	0.688	none	33 epochs
Senet154	0.727	0.740	0.739	44 epochs

Limited crop certain/bitwise/zoom test table (figure 102)

We notice that approach limited crop certain have results that surpass the bitwise normal segmentation method in all models with all epochs this proves that this method is better as we explain in the pre-processing chapter as we said into introduction this approaches bitwise exist only to compare limited crop certain by segmentation method and it proves what discussed.

We notice that depend on approach, number of epochs, and model there is an effect of improved on test time impact.

Each model on different approaches on different epochs have different results we will try to use information that we collect far to explain the behavior in the next sections

After we collect the model, we ensemble it's Metadata by using stacking (average, voting and weighted average) then we will choose the best one to compare to our method Class Weight Transformation

Stacking ensemble models common approaches:

For one particular test sample x, we have M sub-models for the ensemble model. For those models, we have M 7-dimensional vectors named pred_vector_1, pred_vector_2, pred_vector_3, . . ., pred_vector_M. We have tried three common ways to do model ensemble as follows.

1-) Average First best models' ensemble

By using a sum of prediction fractions of each best models then divide it by number of models we could achieve 81.1% as best of the average ensemble, by average equation

$$score = \frac{\sum_{i=0}^{M} pred\_verctor_i}{M}$$

Average Ensemble Test table

approach \ best models	3 models	5 models	7 models
Segmentation bitwise	0.759	0.765	0.774
Limited crop certain	0.789	0.800	0.795
Zoom augmentation	0.806	0.799	0.811

Average Ensemble test table (figure 103)

2-) Voting First best models' ensemble

By using each model as a person that has voted to give away to choose the winning class in prediction, we could achieve 80.3% as best of the voting ensemble. Vote method that takes the max index to convert it to be 1 and make the rest zero in each vector

$$score = \frac{\sum_{i=0}^{M} \text{vote}(pred\_verctor_i)}{M}$$

Voting Ensemble Test table

approach \ best models	3 models	5 models	7 models
Segmentation bitwise	0.760	0.754	0.755
Limited crop certain	0.782	0.785	0.795
Zoom augmentation	0.787	0.796	0.803

Voting Ensemble test table (figure 104)

3-) Weighted Average First best models' ensemble

By using sum in each class prediction fractions of each best model multiply each model prediction by its weight gained from genetic algorithm as an optimizer of weights then divide it by sum of all classes sum, we could achieve 80.9% as best of the average ensemble.

$$score = \frac{\sum_{i=0}^{M} pred\_verctor_i \times \ wieghts_i}{M}$$

Weighted Average Ensemble Test table

approach \ best models	3 models	5 models	7 models
Segmentation bitwise	0.759	0.763	0.773
Limited crop certain	0.776	0.782	0.799
Zoom augmentation	0.809	0.800	0.807

Weighted Average Ensemble test table (figure 105)

But these results mean that normal average did better than weighted average which mean that our genetic algorithm that was fitting using validation data and random weights to do more impact than normal average mean nothing this mean that either we

are overfitting on validation itself or training method that did not use matrices to save model itself is the problem we could not applied weighted on anything even our algorithm itself that have approach to use optimizer, so we will continue using normal methods and compare ours to it, and because we could not use either weighted in both average or our algorithm due this, it is not big problem because we will prove that our method even will surpass even weighted average , so we could continue and set the reason behind this overfitting on validation as future work to continue our research because even if we proved that our method is better we have to explain why and how so we will focus on space of our research and skip explain this part by analysis it.

Class Weight Transformation ensemble

It is an experimental method that we were mentioning too much upside first we have to understand it and explain it, then we will show its results

Class weight is the algorithm that we will show its pseudocode in this section, for one particular test sample x, we have M sub-models for ensemble model. For those models, we have M 7-dimensional vectors named pred_vector_1, pred_vector_2, pred_vector_3, . . ., pred_vector_M as well as weights vector that store each model class's weight

Pseudocode:

Algorithm: Class Weight Transformation

input : weights , softmax-pred
output: new-softmax-pred

```
Begin;
counter = count(weights) − 1;
new-softmax-pred = array[counter+1];
sum = 0;
while counter >= 0 do
    new-softmax-pred[counter] = softmax-pred[counter] × weights[counter];
    sum = sum + new-softmax-pred[counter] ;
    counter = counter − 1;
end
counter = count(weights) − 1;
while counter >= 0 do
    new-softmax-pred[counter] = new-softmax-pred[counter] / sum;
    counter = counter − 1;
end
Return new-softmax-pred;
```

Class weight Transformation pseudocode (figure 106)

$$score = \frac{\sum_{i=0}^{M} Class\_weight(pred\_vector_i, weights_i)}{M}$$

As shown in this equation, class weight takes each predication vector that stores each class prediction probability and multiply each class probability by its class weight for each model then we divide each new probability by sum of all new probabilities for all classes in the single vector.

How can we collect weights of each models?

1-) we can obtain weights as recall of each class from validation

2-) we can obtain weights as f1-score of each class from validation

3-) we can obtain weights as the precision of each class from validation

4-) we can obtain weights from optimizer like genetic algorithm

We will ignore 2 ways to obtain weights 3 and 4 ways, 3 it is precision which we tried and get very bad results even when we tried we expect it as precision does not say that probability of prediction of that class as well as we ignore 4 because we already explain that results from matrices do not have an impact on test time when we used genetic algorithm as an optimizer

Weights as recall of each class from validation

So, recall gained from validation represent class accuracy in our model which is good enough to test it, and we get a score of 82.3% this is improvement more than normal average ensemble.

Recall Transformation Average Ensemble Test table

approach \ best models	3 models	5 models	7 models
Segmentation bitwise	0.766	0.753	0.769
Limited crop certain	0.795	0.804	0.809
Zoom augmentation	0.818	0.823	0.813

Recall Transformation Average Ensemble test table (figure 107)

Weights as f1-score of each class from validation

We could not achieve more than weighted average that is more worsen than normal average we only could achieve 80.4% so we can say that our method is Recall class transformation or genetic algorithm class transformation that may replace it in the future

F1-Score transformation Average Ensemble Test table

approach \ best models	3 models	5 models	7 models
Segmentation bitwise	0.730	0.728	0.729
Limited crop certain	0.775	0.773	0.784
Zoom augmentation	0.804	0.802	0.797

F1 score Transformation Average Ensemble test table (figure 108)

So, if there are good weights then optimizer of weights should find it but we already meet the problem with the weighted average so we skipped that part until we solve it or explain it in the future work.

Recall class transformation

we wanted to test our method in more models more than 7 models in the ensemble, so we must take best 2 approaches models, we chose Zoom and limited crop certain because both have good results compare to the other one but first come first.

We must understand why it works, how it really improve ensemble model itself, to know how or why it happened like this we have to understand that Recall class transformation can be performed as model improvement method it could increase the single model results or decrease it so we understand how does it work but why did works?

We will start by preview its effect on a single model so we will be applied Recall class transformation on each model in those 2 approaches and see test impact.

In this table two up arrows means more than 1% improvement one arrow up mean less than 1% improvement same with down arrows.

Recall Transformation limited crop certain Test table

epochs / model	before	after	effect
Densnet169	0.751	0.753	↑
Densnet201	0.770	0.781	↑↑
Resnet101	0.746	0.744	↓
Resnet152	0.769	0.768	↓
Vgg16	0.744	0.754	↑↑
Vgg19	0.723	0.718	↓↓
Senet154	0.734	0.735	↑

Recall Transformation Zoom Test table

epochs / model	before	after	effect
Densnet169	0.779	0.784	↑
Densnet201	0.776	0.774	↓
Resnet101	0.756	0.773	↑↑
Resnet152	0.770	0.768	↓
Vgg16	0.741	0.738	↓
Vgg19	0.691	0.698	↑
Senet154	0.740	0.745	↑

↑↑ / ↓↓ big effect ——————— ↑ / ↓ small effect

Recall Transformation single model Test table (figure 109)

So in case of that, we use it on the single model it works as adjustor of SoftMax prediction for the single model it moves small fractions based on recall of each class that we gain from validation recall so if we have right class is x1 and x1 has a prediction of 0.495 but x2 has 0.505 then the prediction would say that x2 is the class of sample x so if a class recall of x1 is higher like 80% and x2 has 77% recall if we applied our method would make x1 become 0.50455 and x2 become 0.49544 it change fractions to be better for classes which provide better recall in weights of validation this why some of the models improved and others did not but even got a little worse.

The best model that increased which is resnet101 in zoom approach has a 1.7% increase in test score

So an idea of changing models itself before and keep the best only and ignore model that decreased so we could keep all good models in the normal average might appear in anyone that read this research right now, this idea is good in a point of view of anyone but if we applied it we will notice even if some models go down or decreased that help ensemble model in the end, so after we applied this idea we recognize that this method can be a single model improvement or an ensemble method which its effects help the ensemble model predictions itself even if it lower some models score.

So, let's try to combine models from 2 approaches as we said and compare our method to normal average we already said if our method has better results then there are weights that can improve ensemble then weighted average will be under our method if we used genetic algorithm as an optimizer.

We will test the best 6 models, 10 models and 14 models half of the models in each are from one approach from our main 2 approaches (Zoom and limited crop certain)

Recall Transformation Average Ensemble Test table

Ensemble Type \ best models	6 models	10 models	14 models
Average	0.819	0.829	0.833
Recall Transformation	0.812	**0.834**	0.826

Recall Transformation Average Heavy Ensemble Test table (figure 110)

We notice that our method could even do better than average and with fewer models to get same results plus 0.01%, this is still promising method because it would be even better to use it with optimizer like genetic algorithm with stable models.

Conclusion:

The method class weight transformation or recall transformation proved that it can improve the single model and improve the ensemble between models which can be useful in test time prediction as an adjuster for the probability that came out from SoftMax and in our case could do more than average ensemble with fewer models in the ensemble which mean less flip flops used in computational power needed for it.

This is the method we called it experimental because it just made, in this case, we did not try it with other cases like machine learning algorithms then ensemble by our method in more than one dataset to see effects but this method is worth researching so it is part of our future work for sure so it can be a real function in machine learning which is used to improve models in such cases like ours skin cancer or others cancers and to improve our daily life even more.

As well as it proved that limited crop certain is even better than bitwise by a range of 3% to 5%, the bitwise approach is normal segmentation method by comparing results of both approaches we realize that limited crop certain surpassed bitwise in all models.

Chapter 7 Conclusion and Future Work

Conclusion:

Approach limited crop certain surpass bitwise approach itself, but not all segmentation can work out with all datasets yet our best score was a combination of models in stacking ensemble using our ensemble method which is class weight transformation of both limited crop certain approach and zoom approach yet we did not do any selection of model we just combine best first models for the same reason we could not able to make good weighted average ensemble so our method proved itself in this dataset with results and test time impact.

Future Work:

We have a plan to test our experimental method with different datasets with different models and algorithms to see how far we can apply them on any case or problem, to be common methods to use we need to test it with many cases and build more stable models to test class weight transformation with a genetic algorithm as an optimizer.

We have put plan to do more analysis on class weight transformation behavior in the SoftMax prediction compare it with before and after while compare all that by model training approaches, model losses and redo the same process for single model or algorithm multiple times to see if this effect is fixed or it based on luck like some methods that have random class in it.

We used different common training approaches so when we used weighted average that its weight scored as highest as possible in validation its best test score was less than the best normal average so we have to explain if the train approach or we missed something in the training process with 1 cycle policy. We train without best score matrices to save model because we replace it with best epochs from 3 stages for model due to speed of training that allowed us to train each model in less time than normal so we have to conform reason behind that all matrices that we tested fail to gain a better score. Is it a dataset? Is it a train approaching? Is it something with fastai we missed? Is overfitting on validation because of a genetic algorithm?

We just could test 2 from 4 experimental methods third method called splitter based on the distribution of dataset but it was not so promising on this dataset as we just test it one time, we did not test it with other datasets to confirm that it is hopeless to continue with it and the last method that needs class weight transformation with experimental loss function it is just an idea that we will try to perform in near future.

Chapter 8 Reference

References:

[1] Lozano R, Naghavi M, Foreman K, Lim S, Shibuya K, Aboyans V, et al. (December 2012). "Global and regional mortality from 235 causes of death for 20 age groups in 1990 and 2010: a systematic analysis for the Global Burden of Disease Study 2010" Lancet. 380 (9859): 2095–128.

[2] CDC – Skin Cancer Statistics Archived 8 September 2012 at the Wayback Machine

[3] "Melanoma facts and statistics". Melanoma Institute Australia. Archived from the original on 18 May 2014.

[4] Tschandl, Philipp, Cliff Rosendahl, and Harald Kittler. “The HAM10000 Dataset, a Large Collection of Multi-Source Dermatoscopic Images of Common Pigmented Skin Lesions.” Scientific Data 5.1 (2018)

[5] "Skin Cancer Treatment (PDQ®)". NCI. 25 October 2013. Archivedfrom the original on 5 July 2014.

[6] Gallagher RP, Lee TK, Bajdik CD, Borugian M (2010). "Ultraviolet radiation". Chronic Diseases in Canada. 29 Suppl 1: 51–68. PMID 21199599

[7] Cakir BÖ, Adamson P, Cingi C (November 2012). "Epidemiology and economic burden of nonmelanoma skin cancer". Facial Plastic Surgery Clinics of North America. **20** (4): 419–22. doi:10.1016/j.fsc.2012.07.004. PMID 23084294

[8] Maverakis E, Miyamura Y, Bowen MP, Correa G, Ono Y, Goodarzi H (May 2010). "Light, including ultraviolet". Journal of Autoimmunity. 34 (3): J247-57. doi:10.1016/j.jaut.2009.11.011. PMC 2835849. PMID 20018479.

[9] "Defining Cancer". National Cancer Institute. 17 September 2007. Archived from the original on 25 June 2014.

[10] "General Information About Melanoma". NCI. 17 April 2014. Archived from the original on 5 July 2014.

[11] Leiter U, Garbe C (2008). "Epidemiology of melanoma and nonmelanoma skin cancer--the role of sunlight". Advances in Experimental Medicine and Biology. 624: 89–103. doi:10.1007/978-0-387-77574-6_8. ISBN 978-0-387-77573-9. PMID 18348450.

[12] Chiao EY, Krown SE (September 2003). "Update on non-acquired immunodeficiency syndrome-defining malignancies". Current Opinion in Oncology. 15 (5): 389–97

[13] “Melanoma patterns and structures” dermoscopedia. 10 Sep 2018, 19:54 UTC.

[14] “Classification of nevi” dermoscopedia. Without revision

[15] "Basal cell carcinoma." dermoscopedia. 8 Jun 2019, 11:10 UTC.

[16] "Actinic keratosis." dermoscopedia. 6 Jun 2019, 12:56 UTC.

[17] "Bowen's disease." dermoscopedia. 8 Jun 2019, 11:23 UTC.

[18] "Keratoacanthoma." dermoscopedia. 3 Jun 2019, 06:04 UTC.

[19] "Squamous cell carcinoma." dermoscopedia. 8 Jun 2019, 11:25 UTC.

[20] "Solar lentigines." dermoscopedia. 3 Jun 2019, 06:10 UTC.

[21] "Dermatofibromas." dermoscopedia. 3 Jun 2019, 06:19 UTC.

[22] "Vascular lesions." dermoscopedia. 3 Jun 2019, 06:16 UTC.

[23] "Vascular structures." dermoscopedia. 17 Jun 2019, 09:29 UTC.

[24] "Arborizing blood vessels." dermoscopedia. 24 May 2019, 15:20 UTC.

[25] " Milky red globules." dermoscopedia. Without revision

[26] "Glomerular vessels." dermoscopedia. 24 May 2019, 15:25 UTC.

[27] "Linear irregular vessels." dermoscopedia. 24 May 2019, 15:30 UTC.

[28] "Polymorphous vessels." dermoscopedia. 24 May 2019, 15:31 UTC.

[29] " Corkscrew / tortous vessels." dermoscopedia. Without revision

[30] "Crown vessels." dermoscopedia. 24 May 2019, 15:33 UTC.

[31] "Strawberry pattern." dermoscopedia. 24 May 2019, 15:34 UTC.

[32] "String of pearls pattern." dermoscopedia. 24 May 2019, 15:35 UTC.

[33] "Anatomy of normal skin vasculature." dermoscopedia. 30 May 2019, 12:32 UTC.

[34] "Vessels in the tumor micro environment." dermoscopedia. 30 May 2019, 12:35 UTC.

[35] "Comma vessels." dermoscopedia. 24 May 2019, 15:28 UTC.

[36] "Dotted vessels." dermoscopedia. 17 Jun 2019, 09:30 UTC.

[37] "Hairpin vessels." dermoscopedia. 9 Jun 2019, 14:43 UTC.

[38] Gonzalez, Rafael (2008). Digital Image Processing, 3rd. Pearson Hall. ISBN 9780131687288.

[39] Leslie N. Smith, "Cyclical Learning Rates for Training Neural Networks." (2015).

[40] Leslie N. Smith, et al. "Super-Convergence: Very Fast Training of Neural Networks Using Large Learning Rates." (2017). [41] Alex Krizhevsky, et al. "Imagenet classification with deep convolutional neural networks." *Advances in Neural Information Processing Systems*.

[42] Arnab, Anurag et al. "Conditional random fields meet deep neural networks for semantic segmentation: Combining probabilistic graphical models with deep learning for structured prediction". IEEE Signal Processing Magazine 35.
1(2018): 37–52.

[43] Shervin Minaee, et al. "Image Segmentation Using Deep Learning: A Survey." (2020).

[44] Olaf Ronneberger, et al. "U-Net: Convolutional Networks for Biomedical Image Segmentation." (2015).

[45] Vincent Dumoulin, et al. "A guide to convolution arithmetic for deep learning." (2016).

[46] Noel C. F. Codella, et al. "Skin Lesion Analysis Toward Melanoma Detection: A Challenge at the 2017 International Symposium on Biomedical Imaging (ISBI), Hosted by the International Skin Imaging Collaboration (ISIC)." (2017).

[47] Gatta, Carlo et al. "ACE: An Automatic Color Equalization Algorithm". European Conference on Colour in Graphics, Imaging and Vision (CGIV). (2002).
[48] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, “ImageNet Large Scale Visual Recognition Challenge,” International Journal of Computer Vision (IJCV), vol. 115, no. 3, pp. 211–252, 2015.

[49] D. H. Wolpert, “Stacked generalization,” Neural networks, vol. 5, no. 2, pp. 241–259, 1992.
[50] Nozdryn-Plotnicki, Aleksey et al. "Ensembling convolutional neural networks for skin cancer classification". International Skin Imaging Collaboration (ISIC) Challenge on Skin Image Analysis for Melanoma Detection. MICCAI. (2018).
[51] Nils Gessert, et al. "Skin Lesion Diagnosis using Ensembles, Unscaled Multi-Crop Evaluation and Loss Weighting." (2018).
[52] Zhuang, J., et al. "Skin lesion analysis towards melanoma detection using deep neural network ensemble." ISIC Challenge 2018 2 (2018).