Acknowledgement,

I would like to dedicate this work to my wife and son, who means the world to me.

I appreciate my supervisor for pointing me in the right direction, thank God for providing me with an inspired tutor and finally my parents and siblings would encouraged me on.

TABLE OF CONTENT:

LIST OF FIGURES:

Chapter 1

1 INTRODUCTION:

1.1 Introduction to Deep Active Learning Based Approach for Skin Lesion Classification

Computer aid diagnosis CAD, have in recent years made tremendous progress with the advance of deep learning in AI but the domain of medical image diagnosis remains very delicate mainly because of the safety-critical nature of the domain. Therefore, more than ever smarter way of supervision is necessary. This is where Human-in-the-loop and Active Learning for medical image classification comes in. It can be defined as “the study of computer systems that improve with experience” (Burr 2009) and researcher are begin revisiting the Active Learning framework (Budd, Robinson and Kainz 2021 p. 102062). it is a subfield of Machine Learning that lets the models being trained; decide based on some informativeness measure; what data instances from an Unlabeled data pool to query the Oracle (in our case a medical expert) for labels which is added to the model’s training set so the model can achieve optimal performance.

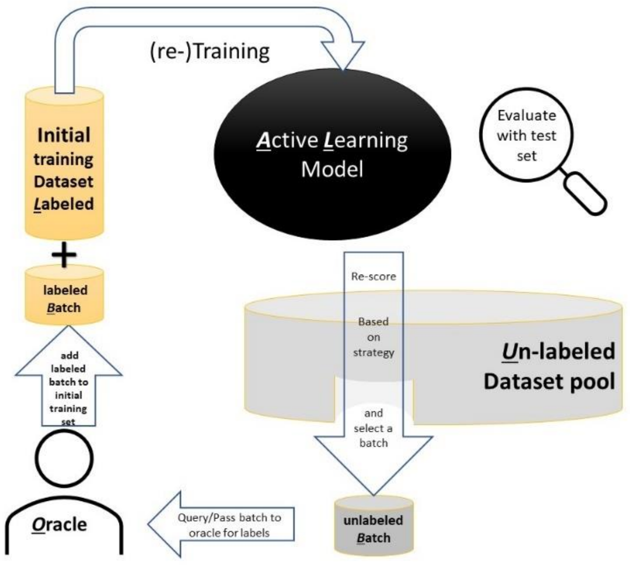


fig 1.1 a basic Active Learn Framework

One way to apply Active learning in Computer Aid Diagnosis models is to create a framework that makes the model query an Oracle (a medical expert) when its diagnosis/prediction are not so confident (i.e., with uncertainty least confident strategy is the criteria used) so that rather than jumping to early conclusion the model will query the medical expert. Such an instance would be deemed Informative to the model because of its low certainty distinguishing between classes (boundary instances) or because the instance is completely new to model (exploratory, more on this chapter2). And when the model is retrained , it should be able to handle similar difficult instances better. Thus, the model has Learnt/Improved which is Active Learning. The key therefore is to design a framework to select this informative instances earlier these is known as the active learning strategy(in theory; it could be that in fact not all the data is relevant for training to obtain the optimal performance, this therefore makes active learning applying to domain where cost of labelling data(data annotation) is relevant so that rather that labelling all the data points in an pool of unlabelled data only the minimum few is present to the oracle and the model can be sure to achieve state-of the art performance) In theory This same Active Learning framework also implies that a when given a little labeled data to start training with and a large pool of un labeled data point, that there may exist a subset of the unlabeled data point that is Informative enough for the model to sufficiently training(after being supplied with their label) to achieve the same or close to same performance as having to get labels and training with the complete un-labeled dataset.

With the increased application of neural network, active learning frame works have seen more and more shifts to deep learning models ; sometimes referred to as deep Active Learning (Wu, Li and Yao 2022 p. 8103)

Medical Images in real world setting come imbalanced, requiring experts' interpretation, noisy with artefacts and acquired under varying conditions (Budd, Robinson and Kainz 2021 p. 102062).

The HAM10000 data set, is a good example of medical image real world data model, the HAM10k data set was selected for this project and it's large collection of 7 classes of skin lesion the dataset characterize with interclass similarity and intraclass difference and very importantly imbalance distribution. The HAM10000 has gone on to become a standard for skin lesion classification challenges.

As Human in the loop computing continues to gain recognition because of medical images sensitivity and The increase in unstructured data which are mostly unlabeled data Active Learning concepts are being explored more for cost effective annotation of unlabeled data in the vast pool of data and in forming critical framework for the deployment of Artificial Intelligence in Computer aided diagnosis

1.2 Hypothesis and problem definition:

What active learning strategy can achieve/surpass baseline performance with as few data instance as possible.

1.3 Chapters description

In the rest of this report I carry out the following:

- Chapter 2 literature Review: Conduct a background survey of crucial and critical literature, exploring key works in Active learning as a whole, it strategies and it's recent application medical images
- Chapter 3 Project Specification: In this section I state the aims and objectives of the project, define the parameters as well as go into the Legal, ethical, social, professional and environmental ramifications of the project.

- Chapter 4 Design and Implementation: describes programmatic the active learning framework, also the designed the of process flow as a data science experiment and describes the design and subsequent implementation of each strategy.
- Chapter 5 Evaluation: In this chapter I present the main results from the three strategies, Random, Uncertainty and Diversity. Then explore in more detail the diversity strategy and the result obtained from diversity with Augmentation a 4th strategy with the aim to review their strengths and weaknesses.
- Future Work: I present the future directions I found most critical and accessible to me.
- Conclusion: present the results and conclusions to the project.

Chapter 2

2 Literature Review

2.1 background literature review.

I cant discuss active Learning without mention the original early Literature survey of (Burr 2009) which he later published into a book, introducing Active Learning before the era of Deep Learn Networks. In his survey Burr describe 3 scenarios (not to be confused with strategies) for Active Learning; the Pool base scenario which is most popular, applicable and I have also adopted in this project mainly because it is less computationally demanding than other scenarios. In pool base scenario; model has access to a pool of unlabeled data, and select the most informative batch another scenario is the stream based scenario where data instances are feed into the model one after another and the model chooses whether to adopt or discard. the third is the synthesis scenario in which new instances are synthesized from real instance and trained on them. Though the pool base is most adopted the rest are still the subjects of some scientific theory based research (Lughofer 2017 pp. 356–376). In his survey burr also identified seven strategies for batch selection *uncertainty, Query by disagreement, query by committee, expected Error Reduction, Variance Reduction, Density-Weighted method and Cluster-based Active Learning*, some of them in practice are extremely hard ware demanding(like the expected Error and Variance reduction) while others like uncertainty and Ensemble strategies saw direct application to Deep Neural Network. while this survey extensive (Monarch and Manning 2021 pp. 89–93) in their book *Human-in-the-loop machine learning Active Learning and Annotation for Human-centered AL* took on a simply, elegantly and updated approach by categorizing the strategies into three groups; Random, Uncertainty/ exploitatory and Diversity/Exploratory. the first Random which is selecting instances randomly is often the baseline for strategy comparisons, Uncertainty/ exploitatory which comprising of least confident, Marginal, entropy and some of the ensemble strategy as well and the Diversity/Exploratory group comprised of cluster base and modal base strategy. Many state of the art deep active model design today are based on uncertainty strategies, and the reason is simply because the output of most classification NN model is a probability distribution, most commonly the "softmax" , which the model back propagates during training. This softmax result fits easily with an uncertainty strategy but as illustrated in their chapter on diversity; the softmax result of models can be vague in as it loses hidden layers activation information. for example referring to fig 2.2, of four instance having same softmax distribution, instance 4. an instance that is on the boundary of four classes high activation across the network and instance 3 that was completely new to the model.

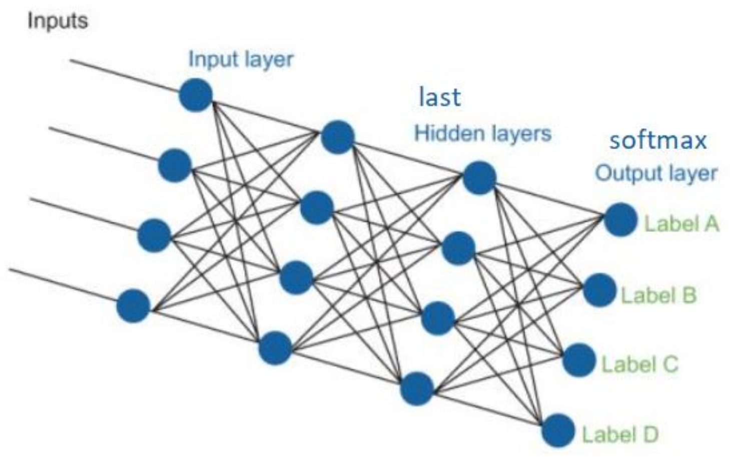
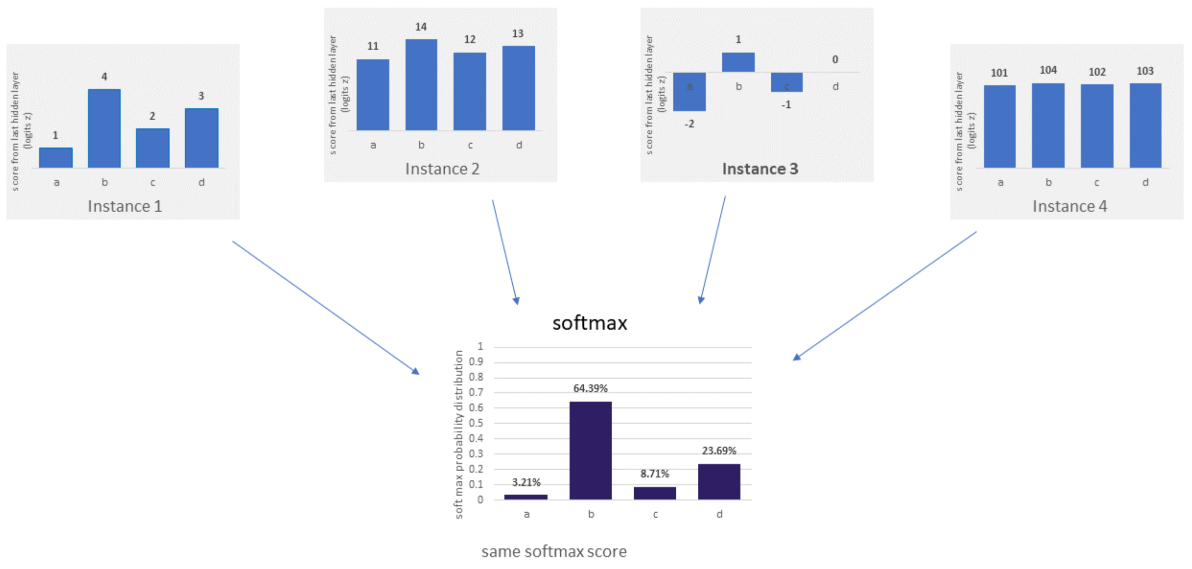


fig 2.1 A simple densely connected 4 class network illustration

as shown by (Monarch and Manning 2021 pp. 89–93)



softmax((1, 4, 2, 3]) = softmax([11, 14, 12, 13]) = softmax([-2, 1, -1, 0|) = softmax([101, 104, 102, 103) = [0.032, 0.6439, 0.0871, 0.2369]

fig 3.2 four identical probability distribution derived from four different inputs/instance

(Monarch and Manning 2021 pp. 89–93)

From the illustration above of this 4 class classification model; we see how the softmax function approximates all four of the instances above to exactly same softmax distribution. one interesting example is the instance 3 which has negative activation in two of the classes, this for example is an instance that model does not have high enough information on ie the activation in those classes for that instance is negative as opposed to the instance 4 were we see very high activation across the 4 classes. From an exploratory perspective instance 3 would be far more informative to a model. In their work they found a way to output the hidden layer activation scores using pytorch API as well as with the softmax score. they were able output this activations called the logitis while still let the models train normally through softmax distribution in its output layer.

This approach, which is a diversity strategy yielded very good performance but consequently for human in the loop this meant that Active learning frame works base on this strategy could detect novel, anomaly and noisy/erroneous instances. Such a diversity strategy could in theory help the model to find this novel and informative instances from the pool early enough and help the model establish ground truth as quicky as possible . the entire implementation in their was base in pytorch API ignoring other powerful Deep learning APIs. also they discussed practical applications this in Human in the loop context but their work was general and not much focus was given to medical Images.

One work that ties Active learning with Skin lesion specifically *is FedAL: An Federated Active Learning Framework for Efficient Labeling in Skin Lesion Analysis* (Deng et al. 2022 pp. 1554–1559) paper. In this work the authors Combines Ensemble uncertainty Entropy base AL with Federated Learning Frame work to achieve very good performance but implemented on a Federated Learning paradigm means having to have a similar Federated Learning system setup.

(Shi et al. 2019 pp. 628–636) employ a dual selective strategy that focuses on informativeness and representativeness(diversity based on PCA nearest neighbors) and interesting add an aggregate augmentation which saw an improvement with the intra class variances their however did not use this in relation to the addressing imbalance in dataset.

In other domains augmentation in Active Learn has been explored (Pop and Fulop 2018) in their work on Deep Ensemble Bayesian Active Learning, show evidence of modal collapse; that is a lack of uniformity of class sampling during data acquiring process of some uncertainty based Active Learning strategies. They also argue that at the early stages of the Active learning process the uncertainty estimation of model is different from the actual uncertainty distribution had the model been exposed to the full data.

Generally Active Learning as part of Human in the loop computing has come a long way seeing diverse application and implementations but as summarize in their survey (Budd, Robinson and Kainz 2021 p. 102062) point out three major challenges DL models faces today in clinical setting:

- the fact that models require large evenly distributed clean and annotated data set to train with. but hardly is this the case in real world scenario.
- the sensitivity of model prediction as checks are needed to double check model output especially in critical and uncertainty conditions and lastly
- transparence of models, interoperability of models is an equally important feature when deployed to real life and critical scenarios

2.2 Motivation

Therefore In this project I intend to show the progress performance behavior of the 3 strategies I test on, to shine the light on the process of training such a network, to show the strength and weakness of the strategies in comparison to each other and as implement a diversity active learning strategy to detect ambiguous and uncertainty instance that are potentially could be of concerning in clinical settings.

Chapter 3

3 Project Specification

In this section I state the aims and objectives of the project, define the parameters as well as go into the Legal, ethical, social, professional and environmental ramifications of the project.

3.1 Aims And Objective

Aim

To design an Active Learning Deep learning strategy, to achieve/approach a benchmark performance in amongst the classes with the HAM10000 dataset achieving this with minimum annotated data points

Objective

- Establish a baseline Models (DenseNet201) on HAM10000 dataset
- Design and experiment with 3 Active Learning Strategy for Dataset implemented with keras
    - Random
    - Uncertainty
    - Diversity
- Review class performance.
- Optimal strategy for Deep Networks for class performance
    - Diversity with Augmentation
- Evaluate results.
- Report strengths and weaknesses of model

3.2 Requirements and scope

Functional requirements

- Design and Implement Active Learning Framework
- Active learning framework should be able to ingest the HAM10000 data set
- Data should be split into required set; test set for evaluation,
- Data should be split into required set; test set for evaluation, initial training and unlabel
- Strategy design should be able to settle batch
- Evaluation Reports should to be generated at each batch addition stage

non-functional requirements

- use of cloud computing environment: google colab pro (premium GPU accelerated) for speed and better performance
- Jupyter notebooks showing result inline
- Access to library for literature
- A Personal Computer for experiment and documentation

The project is a supervised project spanning approximately 12 weeks, a detailed personal **project plan** is included in appendix.

3.3 Methodology

To achieve the aim and objectives of this project I adopted **Scientific Research based methodology;**

- I began by reviewing literature and existing models then **Experiment** (test for and against given hypothesis)
- *ensure experiments are documented and Repeatable
- generate data from experiments
- validate
- conclude.

An alternate methodology would have been the *CRISP-DM* Cross Industry Standard Process for Data Mining which is well suited for data science projects where a data science product is deployed for a particular business need, under the CRISP-DM algorithm selection is flexible, so long it meets a business need. In this project however my focus is on exploring Active Learning Deep Models frameworks hence my selection of the rigorous scientific research methodology

The Data Set:

the HAM10000 data set was selected for this project. This is a large collection of 7 classes (actinic keratoses and intraepithelial carcinoma (AKIEC), basal cell carcinoma (BCC), benign keratosis (BKL), dermatofibroma (DF), melanoma (MEL), melanocytic nevus (NV), and vascular lesions (VASC)) of pigmented skin lesions (Tschandl, Rosendahl and Kittler 2018 vol. 5);

it comprises of demascopic images from diverse populations over the span of 20 years, it was created as a standard for skin lesion classification challenges.

characterized also by imbalance a, inter class similarity and intra class differences the HAM10000 data set mirror well what to expect in a real world medical Image classification challenge.

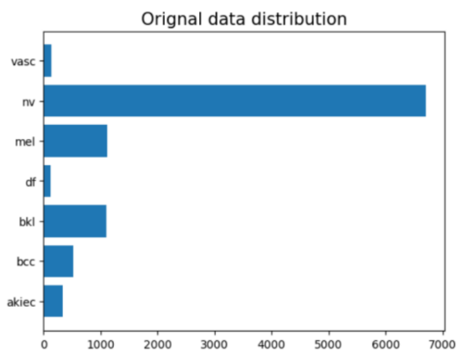


fig 3 1 data distribution of the HAM10000 dataset

project Limitations

- *Deep Learning Api:* In order to meet up with time constraint I kept the project designs in keras API because of it straight forward implementations. other deep learning API I could have worked on are Pytorch, scikit learn, Huggingface , cerat e.t.c
- *Settling on a Baseline Model:* a trade off was reached between performance and time/computation resources available to me. My baseline Model: the pretrained DenseNet201(with 25 retrainable layer) doesn't perform to State of the Art performance seen on the HAM10000 dataset (Sarker et al. 2022 pp. 651–660) because a lot more resources and epochs would be required to retrain the Model at each step. Snice the experiment is focused on Active Learning strategy I could set a relatively easier baseline model seen throughout the experiment while seeing the Active Learning Model under same conditions achieve or approach same results. With 25 unfrozen layers only, the model therefore trains only this layers allowing for much faster trainings.
- *Data Preprocessing:* The HAM10000 dataset is also not preprocessed before passed into the models and no pre augmentation is done to balance the data set. However as part of the Active learning strategy we would later see a post data augmentation in one of the experiment.

3.4 Review of Legal, Ethical, Social, Professional and Environmental Issues and identified risk

As declared before the project began, there are any no ethical issues with the very project

That said, in this section I intend to explore an in depth review of Legal, Ethical, social and Professional issues around Active Learning, Medical Imaging and real world deployment of such systems:

***Legal*:** under the UK legislation this piece of work is cover under copyright laws for Intellectual Property on the other hand all external materials and intellectual property of others have been referenced accordingly. The dataset used, HAM10000 is also under an open source license Attribution-NonCommercial-ShareAlike 4.0 International (CC BYNC-SA 4.0) and the Keras API is also open source therefore the code in this work inherits the open source license which would impact any decision for commercial use. That said, this work is purely a research and academic work.

Ethical and Social: the HAM10000 Data set contains demographic information such as Sex and Age which was not relevant to this project therefore no analysis was conducted on them. Also noteworthy is that racial demographic distribution is not given in HAM10000 dataset, this could impact the ability of the model trained on it to be biased towards the mainly represented skin tone available in the dataset. It was noted however that the data was collated mainly from two sites based in Austria and Australia.

Still discussing Ethics and social issues, the ethics of AI systems and humans in the loop is critical topic today, (Jotterand and Bosco 2020 pp. 2455–2460) in their survey proposed 3 criteria for technology to be integrated into clinical medicine 1. That they serve human ends, 2. they respect personal identity, 3 promotion of human interaction.It can also be argued that AI in medicine or any field should be regulated by professional bodies that regularly review the social impact and state of these technology. Some regulations have begun to come up AL (Goodman and Flaxman 2017 pp. 50–57) *European union regulations on algorithmic decision making and a 'right to explanation'.*

Another social ramification of Active Learning is in the use of Non expert crowd annotators vs fewer Expert Annotator but this is out with the scope of this very project.

Professional: one subtle challenge to Active Learning framework is the fact that the system would require periodic maintenance AL framework if not maintained regularly it could skew toward a biases base on the data input by end user (Budd, Robinson and Kainz 2021 p. 102062) discussed *on How Model could change over time and loss performance on original dataset as they continue to retrain on new batches.*

Finally in accordance with white Paper *World Economic Forum Global Future Council On Human Rights 2016-18, 2018. How To Prevent Discriminatory Outcomes In Machine Learning*; machine learning algorithms are required to be free of systemic biases

This also this work thoughtfully subjects to the four key principles of the BCS institute, RGU being a member have help me understand the benefit of being certified by such a body.

Environmental: Large parts of the experiment were done on google colab research cloud computing platform. Below is computing environment specification:

- GPU: Nvidia V100 GPU or A100 GPU average usage 15/40GB RAM
- SYSTEM RAM 18/83.5 GB
- TOTAL COMPUTE UNITS: About 350
- Total of over 27.5 hours of training
- Region: Europe

The CO_2 Estimated report generated from website https://mlco2.github.io/impact/#publish is:

- CO_2 Emitted 1.44 kg
- Carbon of set by provider 1.44kg

Some other tips followed that contributed to reduced emission were:

- Reduced Training time
- Use of cloud based computing Hardware
- Use of pretrained models Dense net pretrained on “Imagenet”

Risk and mitigants: Other risk includes,

- time constraint for which I tried to stay with project plan successfully without any major setbacks.
- lack of computing resources, move a major part of my work to the cloud reduced the risk of breakdown of my PC, thought the cloud environment required payment in some occasion. Both report and code were backed up on the cloud early on.
- expertise to complete the project, starting the project early with the guidance of supervisor help with this
- The project had no human subjects, no people or animals were hurt or at risks during the course of the project.
- no personal data were collected or stored for this project.

Chapter 4

4 Design and Implementation

In this chapter I start by describing programmatic the active learning framework, also designing the process flow as a data science experiment. I describe the design and subsequent implementation of each strategy as follows:

Section 4.1 I walk through the general framework of the active learning strategies used throughout this work, experiment design in python and data split choices made.

Section 4.2 Random sampling design and implementation.

Section 4.3 Uncertainty strategy (least confidence) design and implementation.

Section 4.4 Diversity strategy design and implementation.

Section 4.5 Diversity with Augmentation design and implementation.

As noted in previous section, this project is implemented in keras because of it simplicity and easy of implementation other more common API for Active learn is Pytorch and scikit learn.

4.1 The general design of the active learning strategies

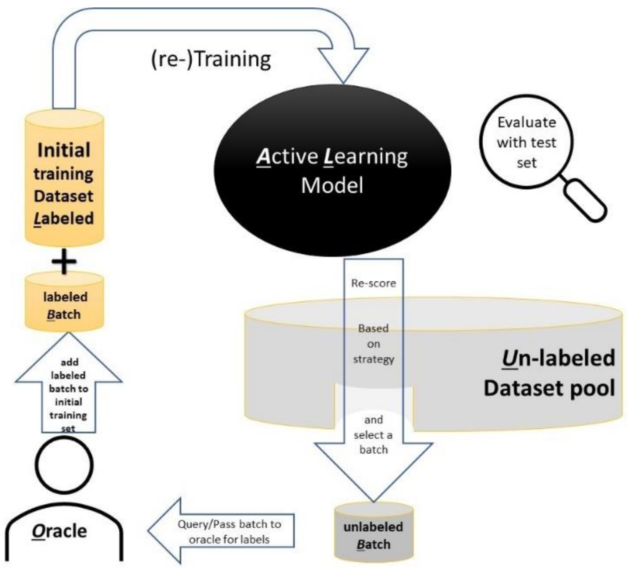


Fig4.1 The general design of the active learning strategies

The model starts with an initial training labeled set typically at 10% of entire dataset, Based on the model’s trained weights and biases, the model will attempt to predict the labels of the an unlabeled pool of data instances, and based on some strategy eg (uncertainty in prediction) it will decide on which instances in the unlabeled pool it found most informative(difficult) then filter down the selection to a set batch amount e.g. 5% of entire dataset to form the batch, it will then consult the oracle for the labels of this selected batch. Once the labels are gotten, the batch is then added to the initial training labeled set.

This framework is designed to test if a strategy can select this informative instance early enough for the Model to achieve similar performance in comparison to when the model is trained with the complete data.

What this means is that theoretically there is a minimum select instance that contain all the information a model needs to train with to perform optimally.

This also means that the oracle which would be a human medical expert, would not be burden with labelling all the data point in the unlabeled pool (cost of annotation).

Also such an effective strategy in real life clinical settings could help the model flag up instance that are new to it and/or confusing, if it was tasked with providing real time diagnosis to large streams of cases/instances

Experiment Design:

The experiment for each strategy was conducted at least 7 time each and averaged out to establish a mean performance and for statistical confidence. In some instances data could not be recorded when the algorithm selected batch sizes slight above required

The experiment where all ran in google colab pro (premium GPU accelerated) under same conditions and in no particular orders

Random seed set is used to ensure reproducibility.

Experiment Data split:

In the creation of the experiment, original HAM10000 dataset is spilt into 0.8/0.2 train/test partitions. Then the training partition is split in to initial Labeled training set(10% of partition) and the rest is stripped of its labels to form Un-labeled pool (90% of training partition). This striped labels are then stored in a variable that can be reference back to for corresponding labels as the Oracle in the case of this experiment,

All the Active learning model and random baseline models starts with the same pretrained model Densenet201 and is train on this 10% initial labeled training set. Then it predicts through instances in the Un-labeled dataset pool and select a batch which gets match with labels from the Oracle and added to the initial labeled training set which at this point will be 15% now and sent to the model for retraining. this batch is removed from the unlabeled pool, so that the Pool continues to diminish and the Labeled set increases through the steps of the Active Learning process flow. At each step the model is evaluated with Test set partition(0.2) (which is unseen to the models or oracle through out retraining steps)

Model design:

the baseline model is imagenet Pretrained Densenet201 with 25 of it's last layers re trainable form the Keras API, two more layers(flatten and dense) and an activation layer were added for the 7 seven classes in the HAM10000 dataset. the baseline results were obtained when the model trains on all the data i.e. at 100% sample ratio for 12 epoch and test on the test set. this is also obtained when the random strategy has sample 100% of the training data

The experiment are designed to compare Active Learning models(based on same architecture as base line model) to the baseline model records.

To maintain fairness all the models are retrained 12 epoch per step(batch) and re-tested on test set to obtain the result recorded, I limited the epoch to 12 because of computational and time constraints on the project.

notation Some active learning terms and notations used throughout the rest of this report, experiment code file and results.

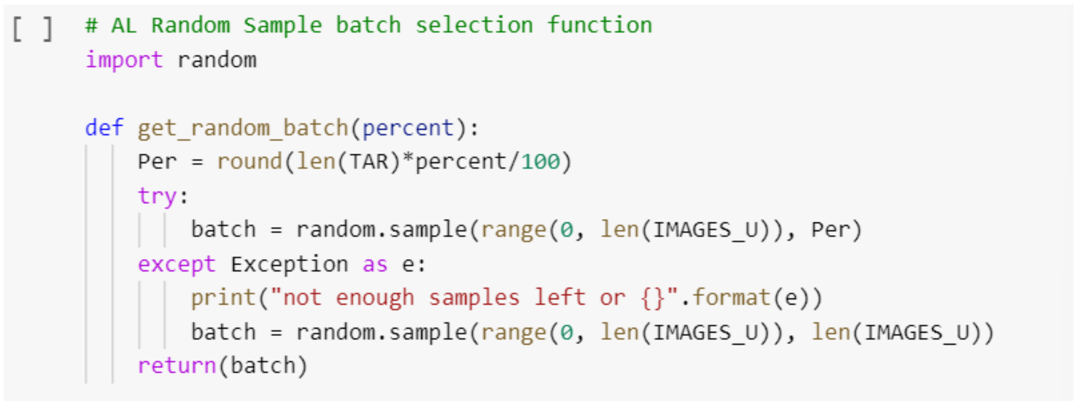
1. Active Learning ***AL***
2. initial labelled training set for training ***IMAGES_L, TARGET_L***
3. Unlabelled pool dataset ***IMAGES_U***
4. based on initial trained model ***ModelAL*** certain batch to be auto-label and absorbed
5. informative batch to be query ***x_batch*** and get ***y*** from ORACLE ***O_Target***
6. add ***x_batch*** to Labelled training set ***L*** (IMAGES and TARGET)
7. remove ***x_batch*** from Unlabelled ***IMAGES_U***
8. retraining model ***ModelAL***
9. testing model ***ModelAL*** on test set
10. reach final model ***M***

through the rest of these sections, I would describe the creation of the Three strategies experimented with and the forth strategy with builds on the third for improved performance

4.2. Random sampling

The random sampling strategy is really the same as baseline model when it reaches 100% of the unlabeled pool U, each training step 10% batch is added to model’s training set

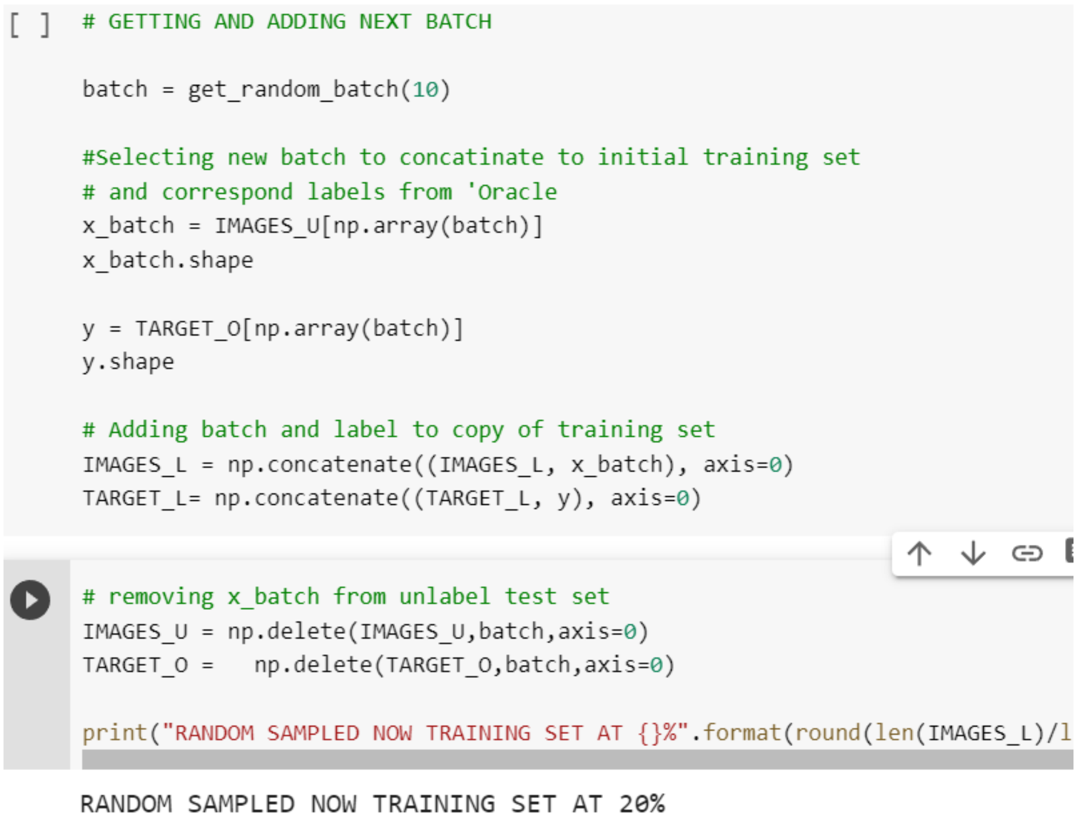
The code below sample randomly indexes of Un labeled pool U



```
[ ]  # AL Random Sample batch selection function
     import random

     def get_random_batch(percent):
         Per = round(len(TAR)*percent/100)
         try:
             batch = random.sample(range(0, len(IMAGES_U)), Per)
         except Exception as e:
             print("not enough samples left or {}".format(e))
             batch = random.sample(range(0, len(IMAGES_U)), len(IMAGES_U))
         return(batch)
```

The next code block uses this index batch to select and remove the batch from the un-labeled pool

Query the Oracle for the labels and add(concatenate) it to the training set



```
[ ]  # GETTING AND ADDING NEXT BATCH

     batch = get_random_batch(10)

     #Selecting new batch to concatinate to initial training set
     # and correspond labels from 'Oracle
     x_batch = IMAGES_U[np.array(batch)]
     x_batch.shape

     y = TARGET_O[np.array(batch)]
     y.shape

     # Adding batch and label to copy of training set
     IMAGES_L = np.concatenate((IMAGES_L, x_batch), axis=0)
     TARGET_L= np.concatenate((TARGET_L, y), axis=0)
```

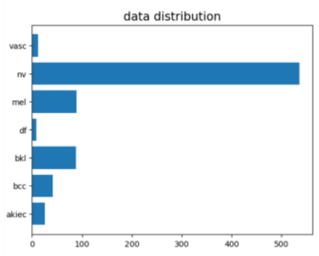
```
     # removing x_batch from unlabel test set
     IMAGES_U = np.delete(IMAGES_U,batch,axis=0)
     TARGET_O =   np.delete(TARGET_O,batch,axis=0)

     print("RANDOM SAMPLED NOW TRAINING SET AT {}%".format(round(len(IMAGES_L)/l
```

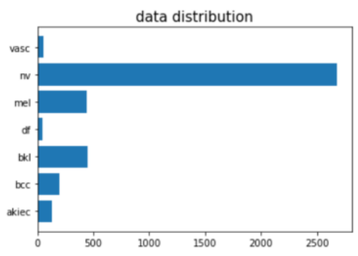
```
     RANDOM SAMPLED NOW TRAINING SET AT 20%
```

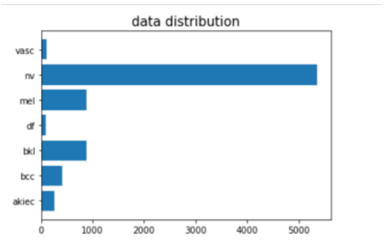
With the random sample the distribution of the classes remained largely the same

With a 10% increment each step until the 10th step where we are at 100% of the data is sampled

The choice of 10% was arbitrary but also convenient so the random sample experiment has only 10 steps







At 10% at 50% at 100%

a **b** **c**

fig4.2 random sampling at different sample ratio

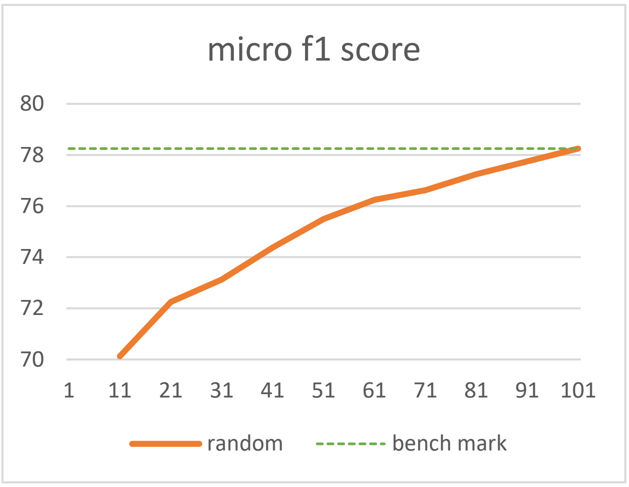


fig 4.3 Random sample gradually reach best performance with increase in data added to training set

4.3 Uncertainty sampling(least confident)

The next strategy call Uncertainty more specifically least confident.

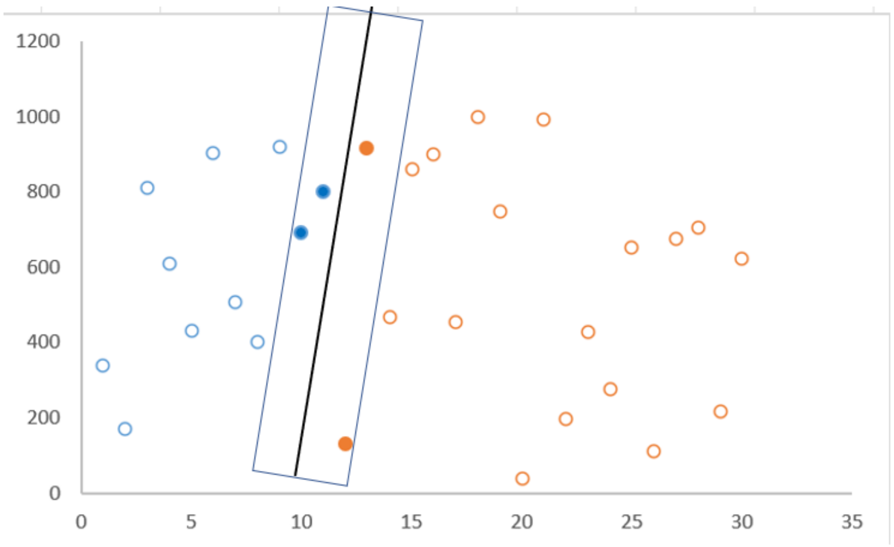


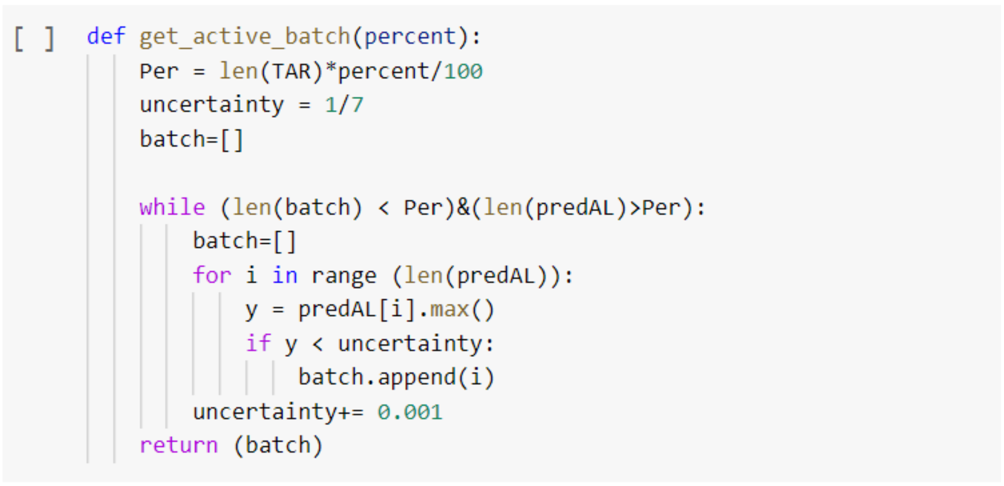
Fig 4.4 uncertainty sampling attempting to find a boundary between two classes.

The Least confidence strategy is type of the uncertainty strategy which includes margin and entropy but least confident unlike the two others is most easily computed. For example in a binary probabilistic model, with an instance that the highest probability score is close to 0.5. specifically in deep models the softmax score for binary classification where the model is least confident would be approximately 0.5 for both classes. Generalizing this for multi-class model is as follows

$$x^*_{LC} = argmax_x\ 1 - P_\theta(\hat{y}|x)$$

Where $\hat{y} = argmax_y\ P_\theta(y|x)$ which is the class with the highest probability for that instance from (Burr 2009)

In this project, I simply start by ranking and selecting instance from most least confident and gradually increasing the confidence level to allow for more instance to fit in my batches.

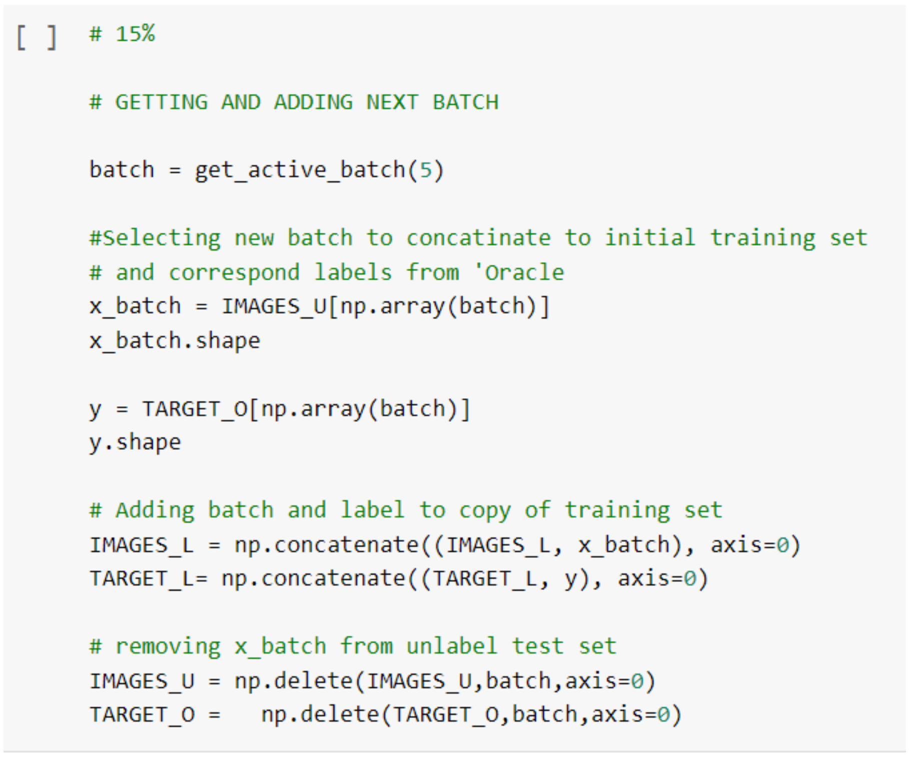


```
[ ]  def get_active_batch(percent):
         Per = len(TAR)*percent/100
         uncertainty = 1/7
         batch=[]

         while (len(batch) < Per)&(len(predAL)>Per):
             batch=[]
             for i in range (len(predAL)):
                 y = predAL[i].max()
                 if y < uncertainty:
                     batch.append(i)
             uncertainty+= 0.001
         return (batch)
```

A percentage for volume of sample to return is supplied and the uncertainty function selects starting from the most least confident i.e. (highest probable class is close to 1/7 of the seven classes) as threshold and gradually continues to increase thes threshold till the number of samples in the batch is approximately equal to or greater than the percentage required. Then the index of this most least confident sample is returned.

Through this strategy I selected a 5% approximate batch size increment because at 4% and above sample rate the model seemed less affected by randomness, also 5% was most convenient to the experiment so not have too many steps, typically 10 steps of 5% was enough.

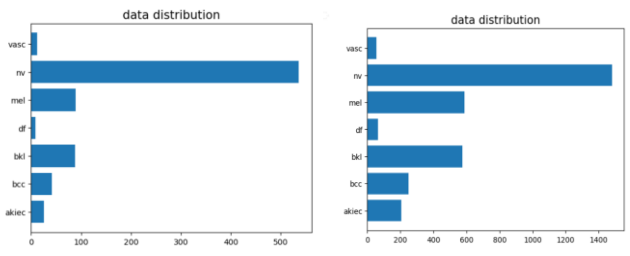
Similarly to how we saw in random sampling above the batch is added to the training set and removed the unlabeled pool



```
[ ]  # 15%

     # GETTING AND ADDING NEXT BATCH

     batch = get_active_batch(5)

     #Selecting new batch to concatinate to initial training set
     # and correspond labels from 'Oracle
     x_batch = IMAGES_U[np.array(batch)]
     x_batch.shape

     y = TARGET_O[np.array(batch)]
     y.shape

     # Adding batch and label to copy of training set
     IMAGES_L = np.concatenate((IMAGES_L, x_batch), axis=0)
     TARGET_L= np.concatenate((TARGET_L, y), axis=0)

     # removing x_batch from unlabel test set
     IMAGES_U = np.delete(IMAGES_U,batch,axis=0)
     TARGET_O =   np.delete(TARGET_O,batch,axis=0)
```

Data sampling

As the uncertainty strategy progresses I also plot the training set data distribution



At 10% at 40%

a **b**

fig4.5 Uncertainty sampling at different sample ratio

4.4 Diversity sampling :

This strategy actually combines both diversity and uncertainty but by prioritizing diversity and progressively adding uncertain instance after exhausting the diverse instances as I will show below. But first a look at the diversity strategy up close:

Neuron activations across a network and softmax activation.

During training or predictions, instances are inputted into the NN model and as the instance is passed deeper into the network if trained on similar classes it will produce high activations in those classes. In the output layer which in most class is conveniently activated using softmax , this activation are transformed into probability distributions and then compare with the actual label and back propagated in the case of training or just outputted in the case of prediction.

As we have seen the output layer (softmax) is curial for the classification algorithm backpropagation model train but by design it looses information on the levels or polarity activation in the network layers by forming it's probability scores

The diversity sample is designed to focus on instance the model has less information on more specifically instances where the model has low activation across the class. To further illustrate this again; take a classifier that is trained on two skin lesion classes dermatofibroma (DF) and melanocytic nevus (NV) images, when it encounters the unclassified unseen image show in fig4.5 b , model will still make a prediction (NV or DF) but notice generally the network had negative scores for that instance.

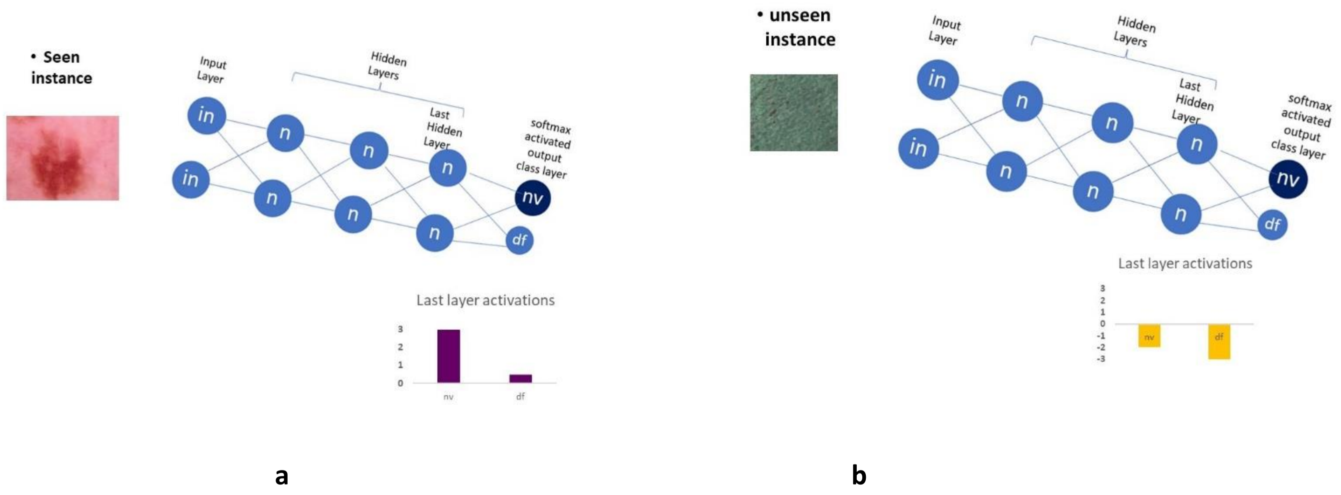
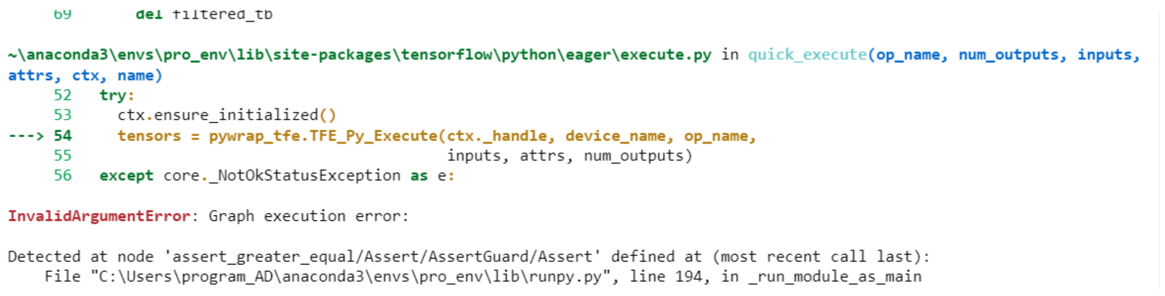


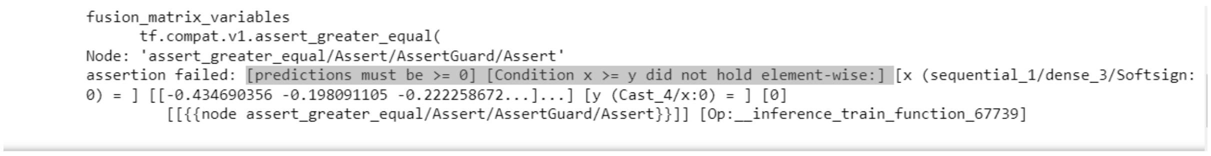
fig 4.6 Illustration of the a binary network train for classification encountering an unseen image

This is where softmax and the last hidden layer logits scores come to play

By extracting this last hidden layer logits scores and ranking from those with most negative activation (most novel instance in this illustration a bird image) we can rank and batch this instance as most informative. In their work on this, (Monarch and Manning 2021) implemented it using the pytorch API, in keras however I was unable to pull this logits scores because the output of models as required by keras has to be positive and between 0-1 for the model to train properly

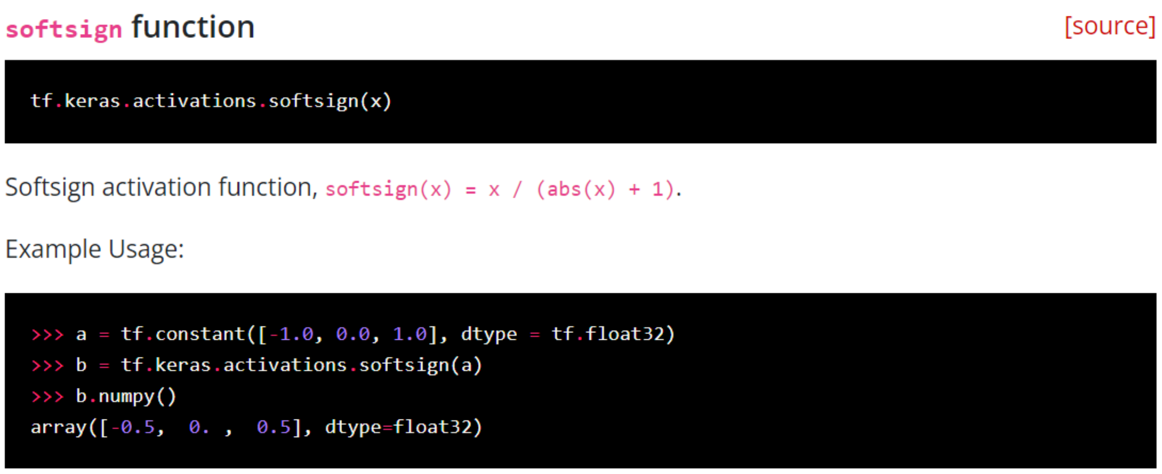


```
     69      del filtered_tb

~\anaconda3\envs\pro_env\lib\site-packages\tensorflow\python\eager\execute.py in quick_execute(op_name, num_outputs, inputs,
attrs, ctx, name)
     52   try:
     53     ctx.ensure_initialized()
---> 54     tensors = pywrap_tfe.TFE_Py_Execute(ctx._handle, device_name, op_name,
     55                                         inputs, attrs, num_outputs)
     56   except core._NotOkStatusException as e:

InvalidArgumentError: Graph execution error:

Detected at node 'assert_greater_equal/Assert/AssertGuard/Assert' defined at (most recent call last):
    File "C:\Users\program_AD\anaconda3\envs\pro_env\lib\runpy.py", line 194, in _run_module_as_main
```



```
fusion_matrix_variables
      tf.compat.v1.assert_greater_equal(
Node: 'assert_greater_equal/Assert/AssertGuard/Assert'
assertion failed: [predictions must be >= 0] [Condition x >= y did not hold element-wise:] [x (sequential_1/dense_3/Softsign:
0) = ] [[-0.434690356 -0.198091105 -0.222258672...]...] [y (Cast_4/x:0) = ] [0]
         [[{{node assert_greater_equal/Assert/AssertGuard/Assert}}]] [Op:__inference_train_function_67739]
```

fig 4.7 Error generated when using soft_sign activation directly in output layer



softsign function [source]

```
tf.keras.activations.softsign(x)
```

Softsign activation function, `softsign(x) = x / (abs(x) + 1)`.

Example Usage:

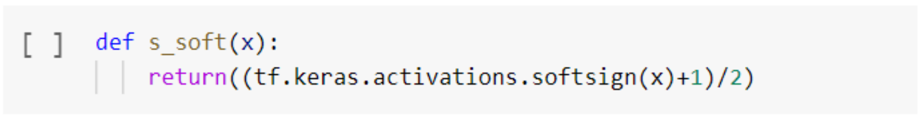
```
>>> a = tf.constant([-1.0, 0.0, 1.0], dtype = tf.float32)
>>> b = tf.keras.activations.softsign(a)
>>> b.numpy()
array([-0.5,  0. ,  0.5], dtype=float32)
```

(Keras 2023)

To solve this problem, I was able to adapt the softsign activation function in keras in my output layer

The softsign function outputs scores from -1 to +1 which would allow me deduct instance with low activation(negative score)

to allow me use this as my output activation layer and still maintain the keras output layer condition I simply scaling the softsign function so that when it out scores I can easily re-scale and get the signs back.



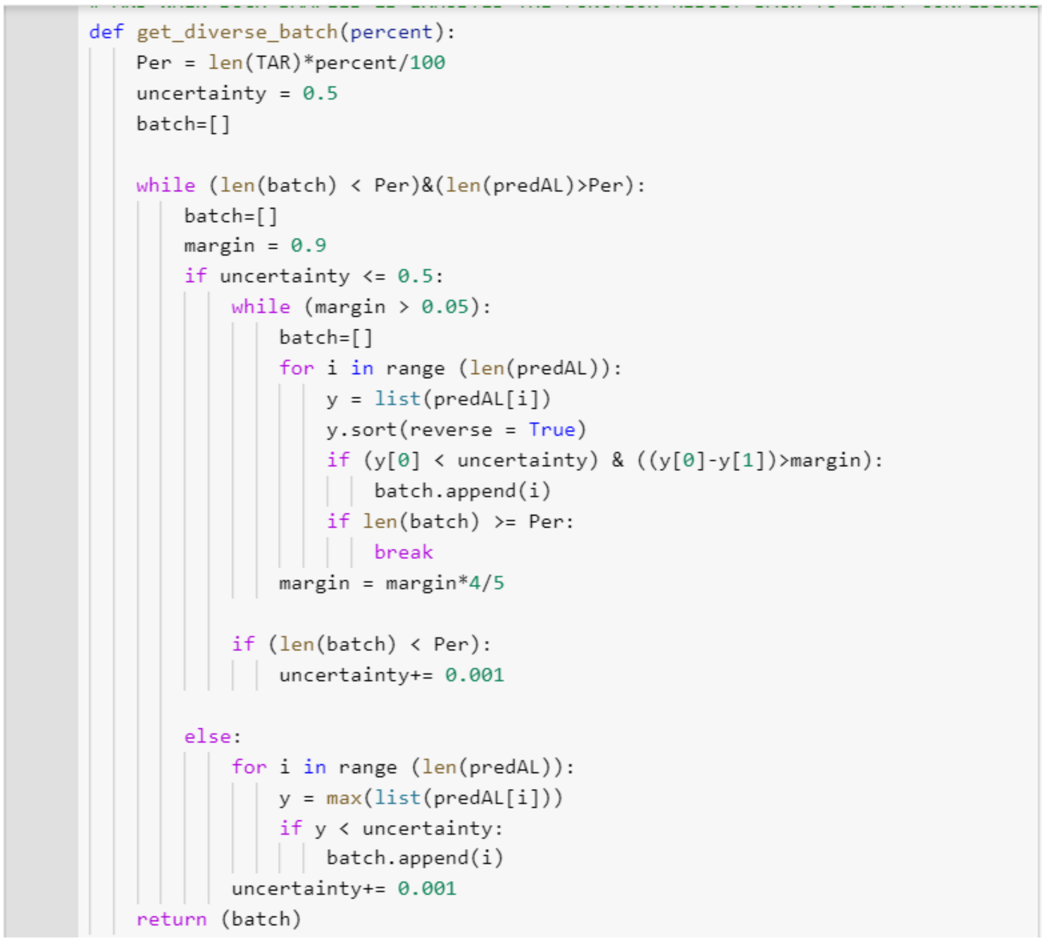
```
[ ]  def s_soft(x):
       return((tf.keras.activations.softsign(x)+1)/2)
```

S_SOFT activation is and adapted activation function that enables me deduct activation polarity of instance predictions. The "softmax" creates a probability representation from 0 to 1 on predicted instance while "softsign" show similar probability but allowing -ve activation to maintain sign. The s_soft activation scaling the softsign function by adding 1 to remove any negative output and dividing by 2 so that -1 equivalent to 0, 0 equivalent to 0.5 and 1 is 1**.**

Define the diversity function while incorporating uncertainty

1. with the s_soft activation: from our prediction on the unlabeled class, see can easily see and deduce that any value below 0.5 was negative

2. It exhaustive select this diverse instances
3. then proceed to fetching in addition least confident (uncertainty samples) next
4. this way this strategy function focuses on this novel instances but still incorporates the uncertainty strategy



```
def get_diverse_batch(percent):
    Per = len(TAR)*percent/100
    uncertainty = 0.5
    batch=[]

    while (len(batch) < Per)&(len(predAL)>Per):
        batch=[]
        margin = 0.9
        if uncertainty <= 0.5:
            while (margin > 0.05):
                batch=[]
                for i in range (len(predAL)):
                    y = list(predAL[i])
                    y.sort(reverse = True)
                    if (y[0] < uncertainty) & ((y[0]-y[1])>margin):
                        batch.append(i)
                    if len(batch) >= Per:
                        break
                margin = margin*4/5

            if (len(batch) < Per):
                uncertainty+= 0.001

        else:
            for i in range (len(predAL)):
                y = max(list(predAL[i]))
                if y < uncertainty:
                    batch.append(i)
            uncertainty+= 0.001
    return (batch)
```

And just as been show in the earlier strategy, the batch is added to the training set and removed the unlabeled pool. The model then retrains and is tested.

4.5 Diversity sampling with Augmentation

Following the Diversity sampling explored above,

This next strategy attempts to incorporate data augmentation to balance the classes by selecting at each step a particular class in the labeled training set to augment.

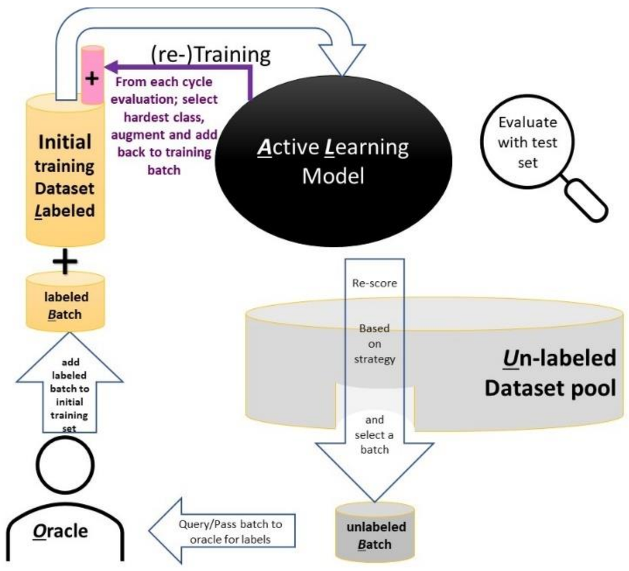

(re-)Training
From each cycle evaluation; select hardest class, augment and add back to training batch
Initial training Dataset Labeled
Active Learning Model
Evaluate with test set
labeled Batch
add labeled batch to initial training set
Re-score Based on strategy and select a batch
Un-labeled Dataset pool
Oracle
Query/Pass batch to oracle for labels
unlabeled Batch

fig 4.8 Modified Active learning Framework to include augmented class

Active learning from uncertainty to the diversity strategy maybe seen to be indirectly/unknowingly balances the class through is sampling. But this is not actually the case. Each Active Learning strategy is only on the look out for informative instances but it turns out in most cases that this informative instance would be in classes that were low in the original training distribution

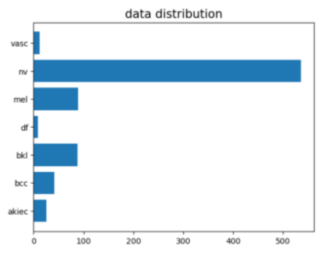


fig 4.9 original initial training distribution

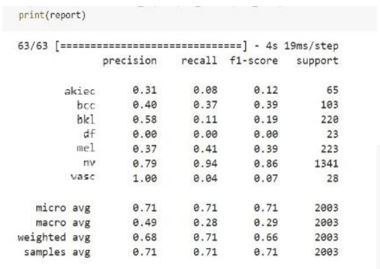
For example virtually all the classes are under sampled except NV class in our case.

Data augmentation up sampling for deep networks attempts to solve the problem of overfitting in small training data during training which is results in poor performance during testing   (Shorten and Khoshgoftaar 2019 pp. 1–48).

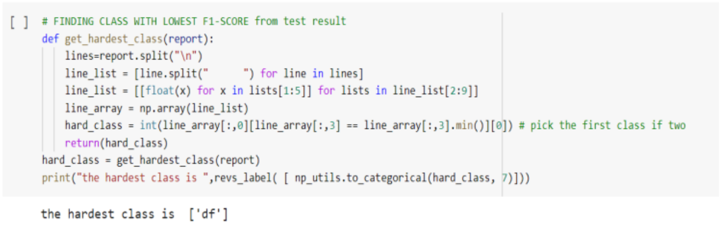
The idea behind this strategy is to actively select the class the model performed less in (from the testing after each batch) and augment that class to see if that class improves faster and to see a general boast in model performance.

Diversity with Augmentation:
The strategy follows

- The model is trained starting with the initial 10% training set.
- Then it determines from the testing performed after each batch selection step which class it found most difficult based on the F1-sorce from sklearn.classification_report



```
print(report)

63/63 [==============================] - 4s 19ms/step
              precision    recall  f1-score   support

       akiec       0.31      0.08      0.12        65
         bcc       0.40      0.37      0.39       103
         bkl       0.58      0.11      0.19       220
          df       0.00      0.00      0.00        23
         mel       0.37      0.41      0.39       223
          nv       0.79      0.94      0.86      1341
        vasc       1.00      0.04      0.07        28

   micro avg       0.71      0.71      0.71      2003
   macro avg       0.49      0.28      0.29      2003
weighted avg       0.68      0.71      0.66      2003
 samples avg       0.71      0.71      0.71      2003
```

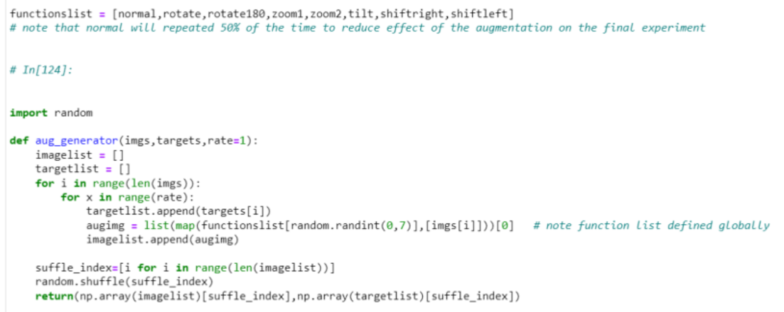


```
[ ]  # FINDING CLASS WITH LOWEST F1-SCORE from test result
     def get_hardest_class(report):
         lines=report.split("\n")
         line_list = [line.split("      ") for line in lines]
         line_list = [[float(x) for x in lists[1:5]] for lists in line_list[2:9]]
         line_array = np.array(line_list)
         hard_class = int(line_array[:,0][line_array[:,3] == line_array[:,3].min()][0]) # pick the first class if two
         return(hard_class)
     hard_class = get_hardest_class(report)
     print("the hardest class is ",revs_label( [ np_utils.to_categorical(hard_class, 7)]))

     the hardest class is  ['df']
```

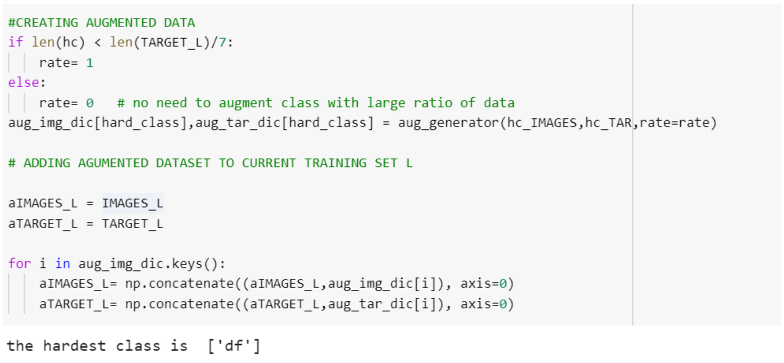
fig 4.10 Report after 1st run at 10% sample ratio

- Then it selects a diversity batch as normal
- Augments the hardest class available in training dataset by a multiple of 1 and add back to batch and initial training set for next step training and testing.

In this project I employ geometric transformation such as rotations, tilt, flips and zoom through an augmentation generator function



```
functionslist = [normal,rotate,rotate180,zoom1,zoom2,tilt,shiftright,shiftleft]
# note that normal will repeated 50% of the time to reduce effect of the augmentation on the final experiment


# In[124]:


import random

def aug_generator(imgs,targets,rate=1):
    imagelist = []
    targetlist = []
    for i in range(len(imgs)):
        for x in range(rate):
            targetlist.append(targets[i])
            augimg = list(map(functionslist[random.randint(0,7)],[imgs[i]]))[0]   # note function list defined globally
            imagelist.append(augimg)

    suffle_index=[i for i in range(len(imagelist))]
    random.shuffle(suffle_index)
    return(np.array(imagelist)[suffle_index],np.array(targetlist)[suffle_index])
```



```
#CREATING AUGMENTED DATA
if len(hc) < len(TARGET_L)/7:
    rate= 1
else:
    rate= 0   # no need to augment class with large ratio of data
aug_img_dic[hard_class],aug_tar_dic[hard_class] = aug_generator(hc_IMAGES,hc_TAR,rate=rate)

# ADDING AGUMENTED DATASET TO CURRENT TRAINING SET L

aIMAGES_L = IMAGES_L
aTARGET_L = TARGET_L

for i in aug_img_dic.keys():
    aIMAGES_L= np.concatenate((aIMAGES_L,aug_img_dic[i]), axis=0)
    aTARGET_L= np.concatenate((aTARGET_L,aug_tar_dic[i]), axis=0)

the hardest class is  ['df']
```

- The model is trained on this training set+ batch + augmented class (e.g. DF or MEL class)

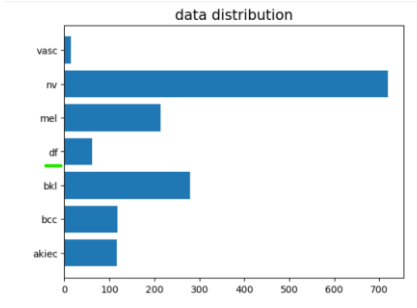


fig 4.11 At 15% with augmented 'DF' class at double the sample amount

- Evaluated on test set

The result and performance of this and the other strategies described above are evaluated and discussed in the next chapter

Chapter 5

5 Final Results and Discussion

In this chapter begin by present the main results from the three strategies, Random, Uncertainty and Diversity. Then explore in more detail the diversity strategy and the result obtained from diversity with Augmentation strategy variation with the aim to see their strengths and weaknesses.

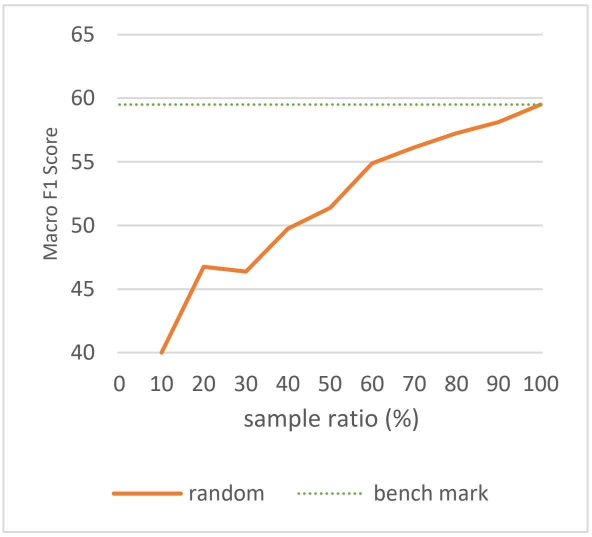
Metrics selection:

The major metric used is the **Macro F1-score** as well as the macro precision and recall because the data set is largely imbalance with melanocytic nevus (NV) class alone contributing 67% out of the 7 classes. Weighted average and micro metric as well as accuracy are heavily skewed by the NV class. The macro metric however averages out all the performance from the all classes better reflecting interclass improvement.

5.1 Main Results:

My three main strategies are, random sampling, uncertainty(least confident) and the Diversity strategy are presented here, with the diversity strategy showing the best average results. At the very earliest stages (before 25%) though, uncertainty showed the most promising result but beyond that point the Diversity strategy out performed both random and uncertainty approaching the benchmark 40% sample ratio.

Fig Random sampled gradually increase sample approaching benchmark at 100%



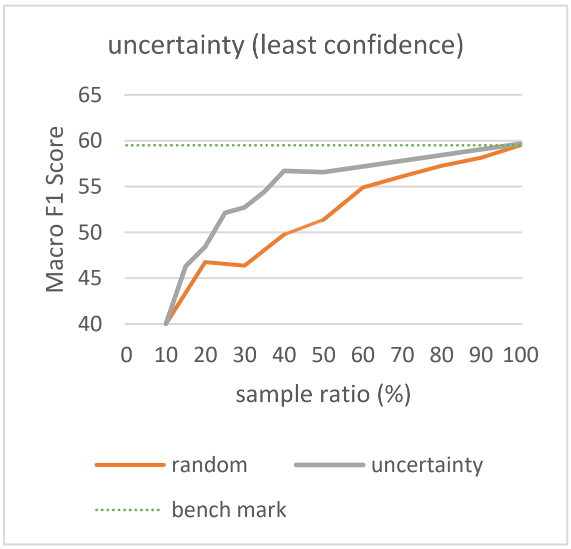


Fig5.1 Comparing the Uncertainty strategy with random sampling

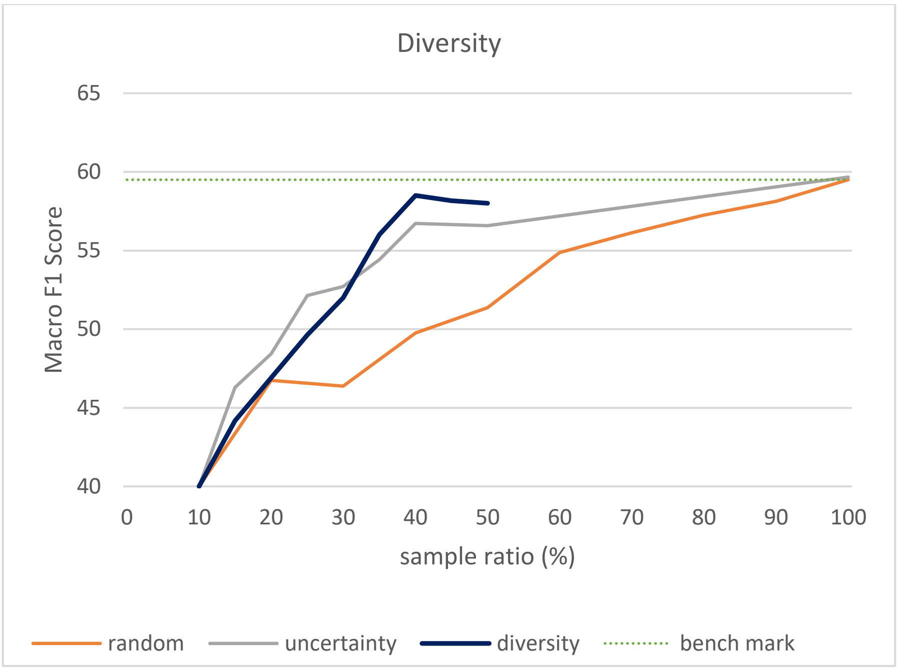


Fig5.2 Comparing Diversity strategy with uncertainty and random sampling

The first graph above shows the progress an Active model makes through its random sampling starting at 10% until it achieves it best performance at 100% of the data.

Similarly, the next graph shows how uncertainty (least confidence) very early on just 40% reach performance levels 95% of the baseline and then stabilizes.

Finally, the 3rd graph showing the diversity strategy (diversity + uncertainty) though starting slowly but at 40% also reach 98% of the baseline. Beyond the 40%, the performance of both the uncertainty and diversity strategy appears to plateau, hence they was not need to keep testing beyond 50%.

Confidence of results:

At 40% sample ratio for both diversity and uncertainty, I performed a confidence test (t-test) from data collected from 7 runs to conclude that with a 95% confidence level; Diversity strategy's average result was higher than that of uncertainty and in the same vein conclude Diversity strategy at 40% reaches 98% of the baseline while that uncertainty reached 95%

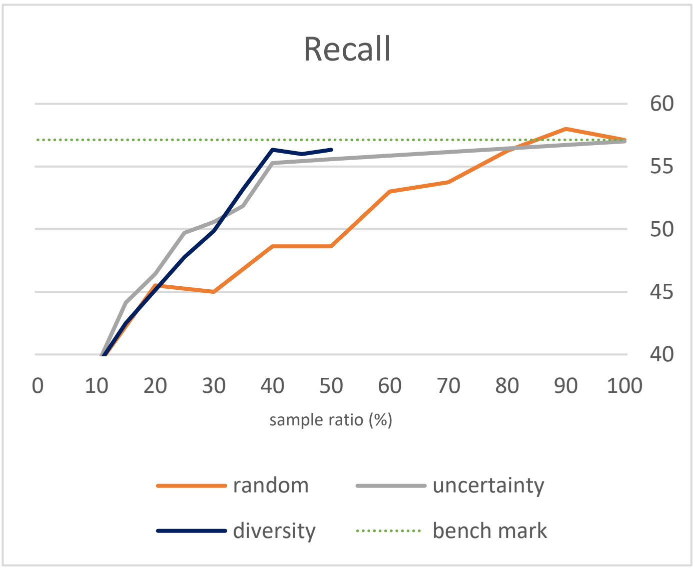
```
        Welch Two Sample t-test

data:  DIVERSITY and UNCERTAINTY
t = 1.5127, df = 11.652, p-value = 0.0785
alternative hypothesis: true difference in means is greater than 0
95 percent confidence interval:
 -0.3882634        Inf
sample estimates:
mean of x mean of y
 58.85714  56.71429
```

Results form a t-test comparing diversity and uncertainty strategy at 40%

5.2 Further evaluations of Diversity strategy

Recall

On of the major factors that dampen the diversity strategy performance at points below 35% was its poor recall, as seen below



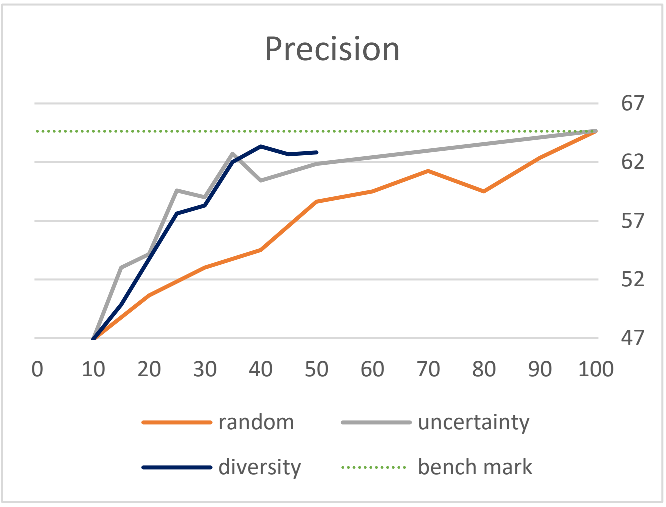


Fig5.3 recall and precision comparison between the 3 strategies

At 20% sample ratio the recall performance was still equivalent to that of random sampling. Though we start to see improvement at 35-40%. With precision on the other hand uncertainty as some point performed better than diversity but both showed closed performance mostly

again it can be generalized that the uncertainty performance at early stages (before 30%) is better that random and the diversity strategy

One technique though used to boast recall has been addition of data augmentation which lead to the design of the Diversity with augmentation strategy described in the previous chapter on design and implementation.

5.3 Diversity with Augmentation: addition of augmentation for boasted performance at early stage.

This strategy as described in the previous section selects a class(poorest performing class) in the training set and augments it and adds it back to the training set after each batch selection step (while still using diversity as batch sampling strategy)

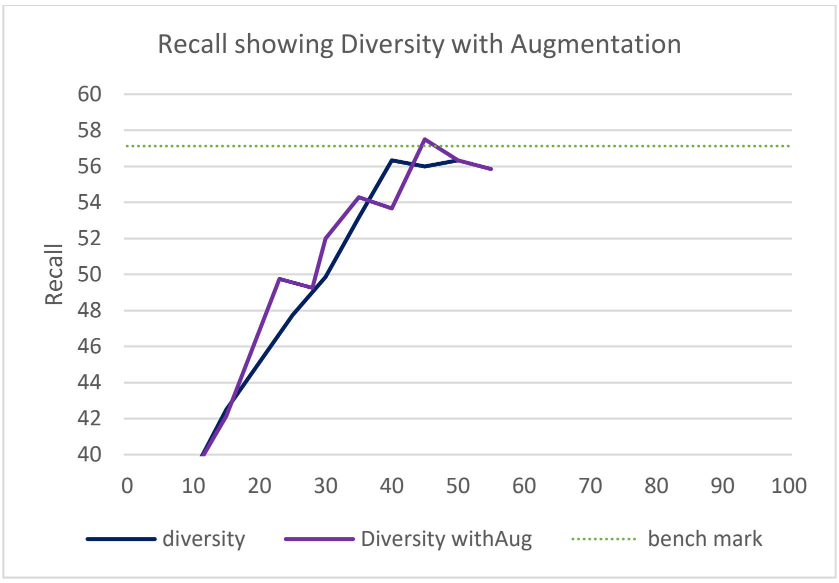
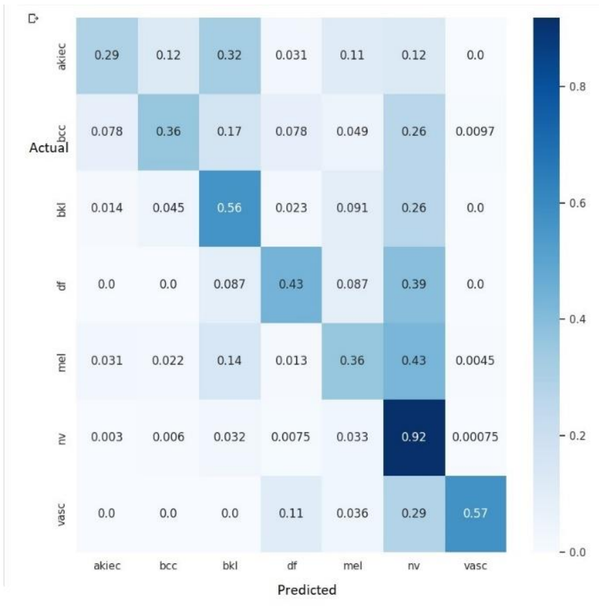


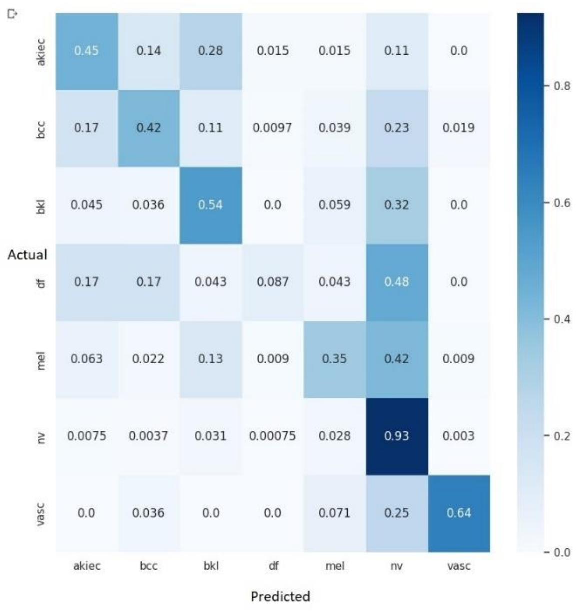
fig 5.4 Recall showing Diversity with Augmentation

Sure enough the augmentation is seen to improve the recall on the strategy, noticeably at the early steps (up till about 25% of the data).

A closer look at the confusion matrix of these two strategies reveals that by augmentation of the DF classes (least performing class at early stages) is seen to boast that class performance and consequently improved recall



	akiec	bcc	bkl	df	mel	nv	vasc
akiec	0.29	0.12	0.32	0.031	0.11	0.12	0.0
bcc	0.078	0.36	0.17	0.078	0.049	0.26	0.0097
bkl	0.014	0.045	0.56	0.023	0.091	0.26	0.0
df	0.0	0.0	0.087	0.43	0.087	0.39	0.0
mel	0.031	0.022	0.14	0.013	0.36	0.43	0.0045
nv	0.003	0.006	0.032	0.0075	0.033	0.92	0.00075
vasc	0.0	0.0	0.0	0.11	0.036	0.29	0.57



	akiec	bcc	bkl	df	mel	nv	vasc
akiec	0.45	0.14	0.28	0.015	0.015	0.11	0.0
bcc	0.17	0.42	0.11	0.0097	0.039	0.23	0.019
bkl	0.045	0.036	0.54	0.0	0.059	0.32	0.0
df	0.17	0.17	0.043	0.087	0.043	0.48	0.0
mel	0.063	0.022	0.13	0.009	0.35	0.42	0.009
nv	0.0075	0.0037	0.031	0.00075	0.028	0.93	0.003
vasc	0.0	0.036	0.0	0.0	0.071	0.25	0.64

Diversity with Augmentation Diversity

Fig5.5 Confusion matrix batch between Diversity_with_Augmentation and Diversity

Model seen to be predicting DF class better at 25% in the Diversity Aug Confusion matrix compare with the diversity Matrix

This also resulted in a boast in f1 score in the early steps see below. But drastic drops in precision at 30% also results in the drop of the over all f1 score at the later steps

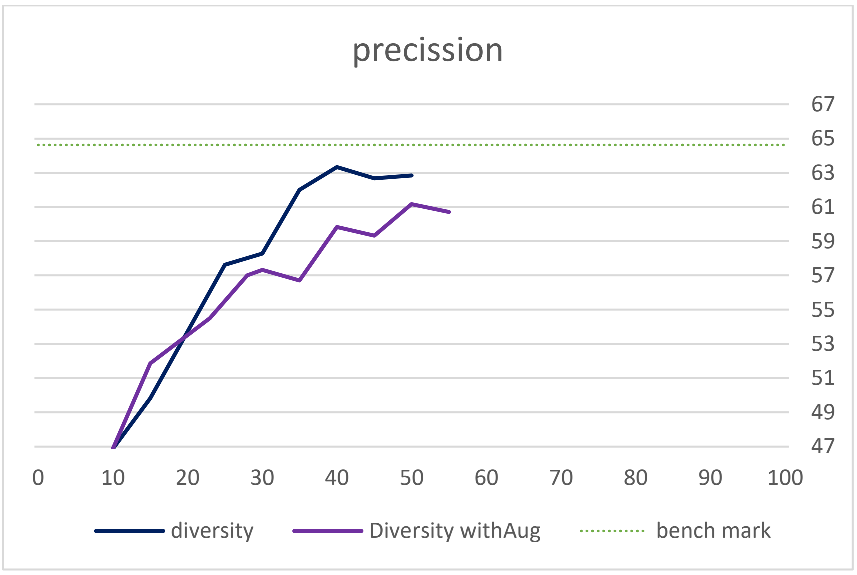


Fig5.6 .4 Precision plot between Diversity_with_Augmentation and Diversity

Overall diversity with augmentation seem like a good idea and in the way it was implemented in this project; produced promising performance at the early stages sampling by boasting recall. this however was only at the early stages leaving room for further investigation. it can therefore be presumed that class augmentation does have a positive effect on a strategy at the early steps of Active learning sampling.

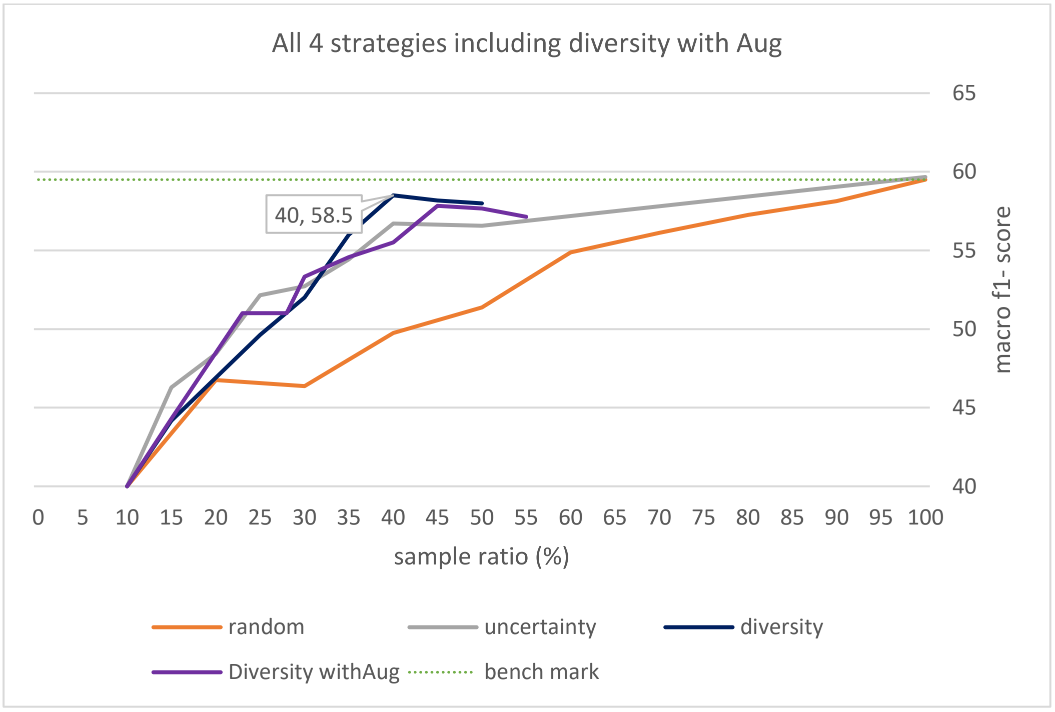


Fig5.7 All 4 strategies including diversity with Aug

5.3 Strategy's Data Distribution Analysis

Another interesting outcome from the experiments ran were Strategy's Sample data distribution behaviors presented below for random sampling, Uncertainty, Diversity and Diversity with Augmentation strategies. Showing the distribution with progressing sample ratios.

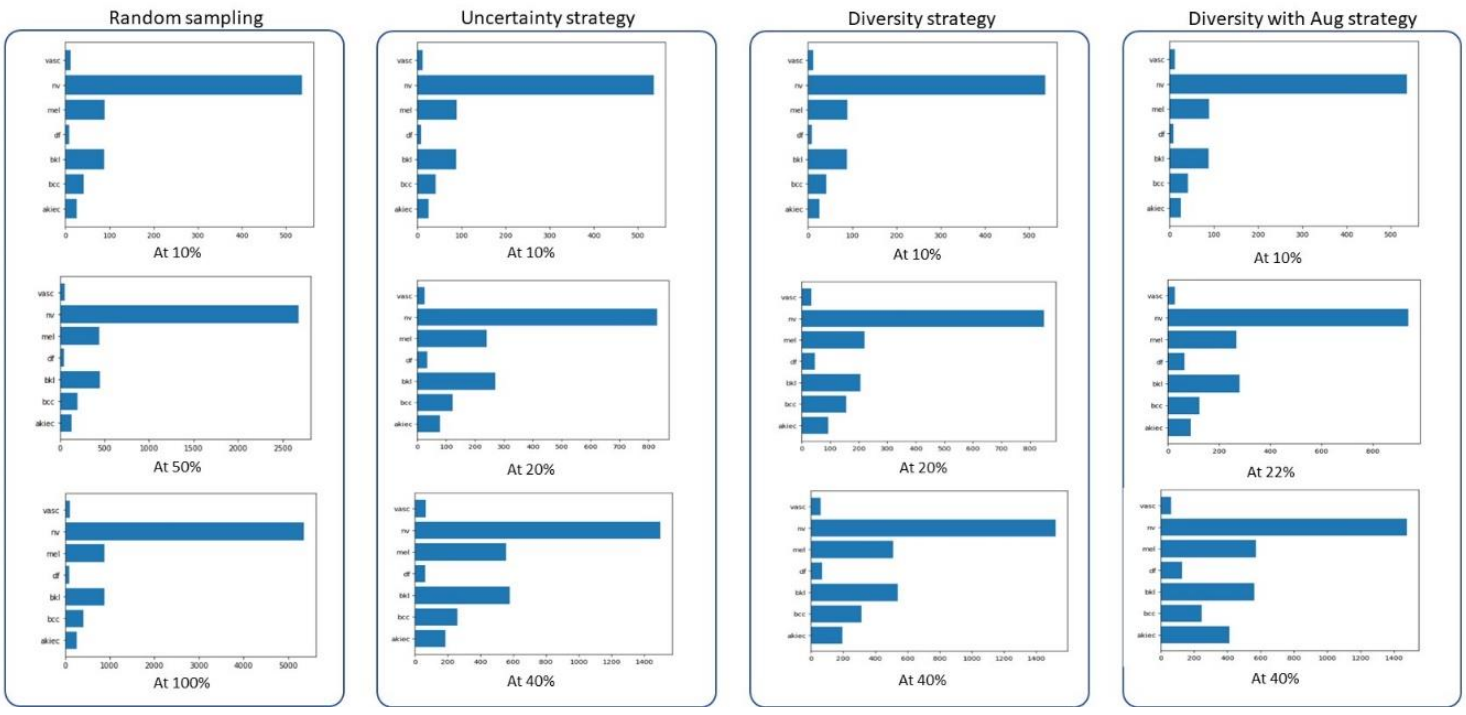


Fig5.8 Data distribution behaviours observed in the 4 strategies

apart from the random sampling, the other strategies as we have seen, samples based on the informativeness criteria of the instances in the unlabeled pool when passed to the Active Learning model, different strategies therefore prioritize and sample certain classes more. looking at uncertainty based on visual comparison we can see that early on it samples a lot the BKL class followed by MEL on the other hand the diversity strategy is seen to spread it's focus amongst the classes Diversity with Augmentation strategy early on doubles the number for DF class and later the AKIEC class.

The effect of these strategies sampling would be better observed and studied with a uniformly distributed data set to reveal the strategies and their biases.

5.4 Critical Evaluation Of Project/ Limitations Of Work And Reflection

In hindsight these were some of the limitations in this project.

When compared with initial aim which was to surpass baseline performance, the Active learning strategies designed didn't surpass but the mean result form the 8 runs came up 98% of the baseline. and there were instance in the 8 runs that actually surpassed the baseline but the average result did not.

- Project Base Keras API: The Keras API alone was used through the course of the project because of it's simplicity and straight forward application in Active Learning but strategies like the diversity described in this work were originally implemented and demonstrated in PYTORCH (Monarch and Manning 2021 pp. 89–93) . I eventually worked around the keras API but acknowledge implemented in the Pytorch would have seen different results.

- Baseline model: The objective was still achieve thought not with a state-of-the-art model but in hindsight random hyper parameter tuning would have been helpful.

- Transferring the active learning strategies used in this project to other data set may perform differently but subject to experimentations though the expectation is that the results should equally be good see how it has held up to the HAM10000 but this is not tested yet

6 Future Work

This project has indeed been enlightening to me and opened several aspects of Active Learning to me, but I would present the directions I found most critical and accessible to me.

- As established active learning doesn't really attempt to balance data set but possibly active learning through its process can provide information as to which classes may need augmentation/up sampling and by what factor. also **finding best use of augmented data to achieve optimal performance. One work that starts to explore this is** (Shi et al. 2019 pp. 628–636)
- Another aspect I would explore are the techniques of transfer learning that would allow the freezing of aspect of the model that have achieve top performance while allow the others to re-train as an attempt to solve the problem of deteriorating performance sometimes seen in at class levels and in the general active learning models performance (Cheplygina, de Bruijne and Pluim 2019 pp. 280–296)
- I would also incorporate practical interactive oracle querying techniques how the actual images are presented to the annotator to maximize cost of annotation (Stember et al. 2019 pp. 597–604)

7 Conclusions.

The active learning framework designed in this work reaches 98% of the baseline at 40% of the train data sample ratio with a 95% confidence level. In this project I also explored the strength and weakness of the strategies and attempted the use of data augmentation improve performance which saw early boost in recall performance but only at the initial stages. Still the use of augmentation techniques is a promising area of future work.

Reference

BUDD, S., ROBINSON, E.C. and KAINZ, B., 2021. A survey on active learning and human-in-the-loop deep learning for medical image analysis, 71, p. 102062. Available from: https://www.sciencedirect.com/science/article/pii/S1361841521001080.

BURR, S., 2009. Active Learning Literature Survey. [online]. Available from: https://burrsettles.com/pub/settles.activelearning.pdf [Accessed 26 Jan 2023].

CHEPLYGINA, V., DE BRUIJNE, M. and PLUIM, J.P.W., 2019. Not-so-supervised: A survey of semi-supervised, multi-instance, and transfer learning in medical image analysis, 54, pp. 280–296. Available from: https://www.sciencedirect.com/science/article/pii/S1361841518307588.

DENG, Z. et al., 2022. FedAL: An Federated Active Learning Framework for Efficient Labeling in Skin Lesion Analysis. pp. 1554–1559.

GOODMAN, B. and FLAXMAN, S., 2017. European Union Regulations on Algorithmic Decision-Making and a “Right to Explanation”, 38(3), pp. 50–57. Available from: https://search.proquest.com/docview/1967052651.

JOTTERAND, F. and BOSCO, C., 2020. Keeping the “Human in the Loop” in the Age of Artificial Intelligence, 26(5), pp. 2455–2460. Available from: https://doi.org/10.1007/s11948-020-00241-1.

KERAS, 2023. *Keras documentation: Layer activation functions*. [online]. Available from: https://keras.io/api/layers/activations/ [Accessed 16 Apr 2023].

LAN, Z. et al., 2022. FixCaps: An Improved Capsules Network for Diagnosis of Skin Cancer, 10, pp. 76261–76267. Available from: https://ieeexplore.ieee.org/document/9791221.

LUGHOFER, E., 2017. On-line active learning: A new paradigm to improve practical useability of data stream modeling methods, pp. 356–376. Available from: https://www.sciencedirect.com/science/article/pii/S0020025517308083.

MONARCH, R. and MANNING, C.D., 2021. *Human-in-the-loop machine learning*. Shelter Island: Manning. Available from: http://bvbr.bib-bvb.de:8991/F?func=service&doc_library=BVB01&local_base=BVB01&doc_number=032881704&sequence=000001&line_number=0001&func_code=DB_RECORDS&service_type=MEDIA.

POP, R. and FULOP, P., 2018. Deep Ensemble Bayesian Active Learning : Addressing the Mode Collapse issue in Monte Carlo dropout via Ensembles. [online]. Available from: https://arxiv.org/abs/1811.03897.

SARKER, M.M.K. et al., 2022. TransSLC: Skin Lesion Classification in Dermatoscopic Images Using Transformers. In: G. YANG et al., eds. Medical Image Understanding and, 2022. Cham: Springer International Publishing. pp. 651–660.

SHI, X. et al., 2019. An Active Learning Approach for Reducing Annotation Cost in Skin Lesion Analysis. In: H.-I. SUK et al., eds. Machine Learning in Medical, 2019. Cham: Springer International Publishing. pp. 628–636.

SHORTEN, C. and KHOSHGOFTAAR, T.M., 2019. A survey on Image Data Augmentation for Deep Learning, 6(1), pp. 1–48. Available from: https://link.springer.com/article/10.1186/s40537-019-0197-0.

STEMBER, J.N. et al., 2019. Eye Tracking for Deep Learning Segmentation Using Convolutional Neural Networks, 32(4), pp. 597–604. Available from: https://doi.org/10.1007/s10278-019-00220-4.

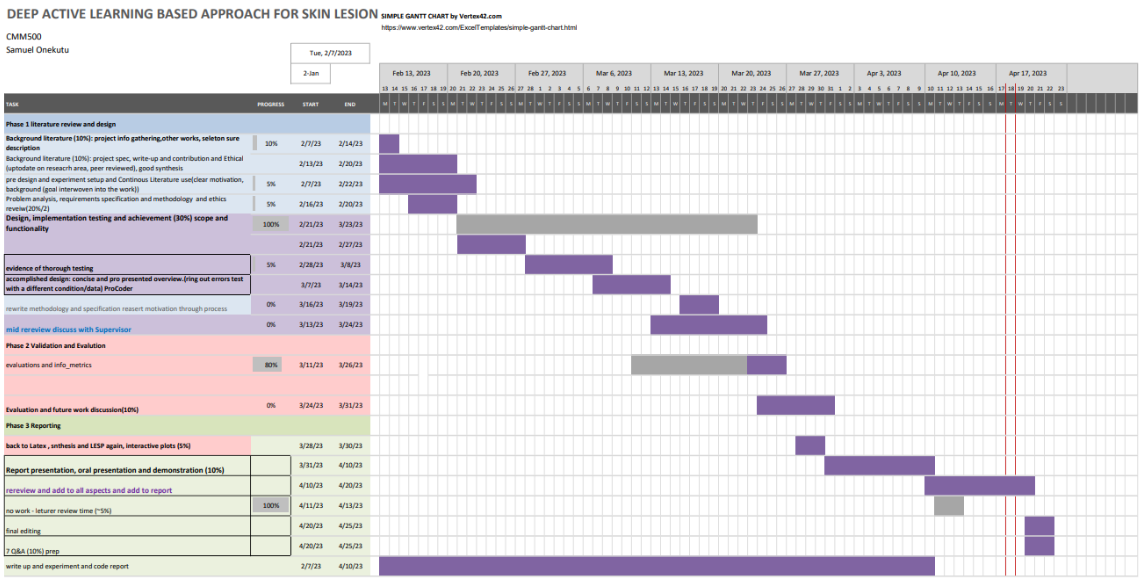
TSCHANDL, P., ROSENDAHL, C. and KITTLER, H., 2018. The HAM10000 Dataset: A Large Collection of Multi-Source Dermatoscopic Images of Common Pigmented Skin Lesions, 5.

WU, M., LI, C. and YAO, Z., 2022. Deep Active Learning for Computer Vision Tasks: Methodologies, Applications, and Challenges, 12(16), p. 8103. Available from: https://search.proquest.com/docview/2706114205.

Appendix A - Personal Project Management

Gnatt chart for Personal Project Management:

Asides from the weekly guidance from my supervisor Dr Carlos Moreno-Garcia, I also had the gantt chart below:



DEEP ACTIVE LEARNING BASED APPROACH FOR SKIN LESION

CMM500

Samuel Onekutu

Tue, 2/7/2023

2-Jan

TASK	PROGRESS	START	END
Phase 1 literature review and design			
Background literature (10%): project info gathering,other works, seleton sure description	10%	2/7/23	2/14/23
Background literature (10%): project spec, write-up and contribution and Ethical (uptodate on reseacrh area, peer reviewed), good synthesis		2/13/23	2/20/23
pre design and experiment setup and Continous Literature use(clear motivation, background (goal interwoven into the work))	5%	2/7/23	2/22/23
Problem analysis, requirements specification and methodology and ethics reveiw(20%/2)	5%	2/16/23	2/20/23
Design, implementation testing and achievement (30%) scope and functionality	100%	2/21/23	3/23/23
		2/21/23	2/27/23
evidence of thorough testing	5%	2/28/23	3/8/23
accomplished design: concise and pro presented overview.(ring out errors test with a different condition/data) ProCoder		3/7/23	3/14/23
rewrite methodology and specification reasert motivation through process	0%	3/16/23	3/19/23
mid rereview discuss with Supervisor	0%	3/13/23	3/24/23
Phase 2 Validation and Evalution			
evaluations and info_metrics	80%	3/11/23	3/26/23
Evaluation and future work discussion(10%)	0%	3/24/23	3/31/23
Phase 3 Reporting			
back to Latex , snthesis and LESP again, interactive plots (5%)		3/28/23	3/30/23
Report presentation, oral presentation and demonstration (10%)		3/31/23	4/10/23
rereview and add to all aspects and add to report		4/10/23	4/20/23
no work - leturer review time (~5%)	100%	4/11/23	4/13/23
final editing		4/20/23	4/25/23
7 Q&A (10%) prep		4/20/23	4/25/23
write up and experiment and code report		2/7/23	4/10/23

Appendix B - Results

RANDOM SAMPLING marco-results	precision	recall	f1-score	micro F1-Score
10	46.87	39	40	70.12
20	50.62	45.5	46.75	72.25
30	53	45	46.37	73.12

RANDOM SAMPLING