A PROJECT REPORT ON

**Data Labelling of URL Using Active Learning**

A project report submitted for Machine learning (CS-584)

Project submitted by

**Shivani Shrivastav & Ankan Mazumdar**

Under the Guidance of Professor: Binghui Wang

Illinois Institute of Technology

Declaration of Authorship

We solemnly affirm that the thesis entitled "Data Labelling of URL using Active Learning " is a genuine work undertaken by the undersigned. This document has been developed in compliance with the established academic norms and principles of ethical conduct. Furthermore, in adherence to these standards, we ensure that all sources and materials not original to this work have been appropriately acknowledged and cited.

**Name:** Shivani Shrivastav & Ankan Mazumdar

**Thesis Title:** Data Labelling of URL Using Active Learning

ACKNOWLEDGEMENT

Every project big or small is successful largely due to the effort of a number of wonderful people who have always given their valuable advice or lent a helping hand. I sincerely appreciate the inspiration; support and guidance of all those people who have been instrumental in making this project a success. We, Ankan Mazumdar & Shivani Shrivastav, are extremely grateful to our professor for the confidence bestowed in us and entrusting my project entitled “**Data Labelling of URL using Active Learning**” with special reference.

At this juncture, I express my sincere thanks to Professor**: Binghui Wang** of Computer Science Department for making the resources available at the right time and providing valuable insights leading to the successful completion of our project who even assisted us in completing the project.

**Contents**

**Abstract**

**Literature Review**

**Problem Statement**

**Data Description**

**Approach**

**Requirement analysis**

**Initial Preprocessing**

**Model Building and training**

**Model improvement strategy**

**Deployment**

**References**

**Abstract:**

Data labeling can be a very time- and/or money-consuming procedure. For this, a domain specialist is occasionally required. Active Learning is an approach which uses fewer training data to achieve better optimization by iteratively training a model. We can use a classification model to perform the majority of the labeling using active learning, requiring us to label samples only when absolutely necessary. Active learning seeks to address this issue by asking to annotate only the most informative data from the unlabeled set.If given the freedom to choose which data to label, the active learning methodology has the potential to significantly reduce the amount of labeled data needed to train a model with improved accuracy. The data that the model is most unsure of is prioritized through active learning, and labels are only requested for those. As a result, the model picks up new information more quickly.

This is a very important problem ,the reason being, in real world situations, we might want to go ahead to obtain manual labels for new data, but we have constraints on how much labeled data we can actually obtain within a certain time & limited budget.

**Literature Review**

Managing and streamlining a data annotation process is difficult. Businesses face a number of internal and external challenges that make the work of annotating documents inefficient and ineffective. Therefore, the only way to overcome these difficulties is to get to the root of them, comprehend them, and then resolve them appropriately. Let's begin.

Having trouble managing a large workforce-

Data-hungry ML and AI models need a huge amount of labeled data to learn. Businesses employ a sizable staff to manually tag datasets in order to provide the vast amount of labeled data needed to feed the computer models. Additionally, processing Big Data and labeling them with the best quality is crucial to achieving a high level of accuracy.

a lack of access to innovative tools and technologies-

Having a large and skilled workforce does not guarantee the production of high-quality tagged datasets. To carry out the accurate data annotation process, the right equipment and technology are needed. Different technologies and methods are used to tag datasets for deep learning depending on the type of data. It is therefore crucial to deploy the right technology that guarantees the greatest quality at a reasonable price.But businesses frequently fall short in creating the infrastructure necessary for the greatest possible data annotation. Because of the high cost of the tools and a lack of professional process expertise, organizations struggle to choose the best technology to implement.

Lack of reliable and high-quality data tagging-

A precise data annotation model necessitates the highest caliber dataset tagging. There is no room for error at all. Small mistakes can have a major impact on the company's bottom line. The ML model will pick up on incorrect information from your data samples in the same way. As a result, AI won't be able to correctly forecast it or recognise it.Furthermore, obtaining consistent high-quality data is the true challenge; assuring it is not the primary criterion. It is essential for organizations to keep a steady flow of high-quality, labeled datasets for machine learning (ML) training and accurate AI prediction.

Not an inexpensive project-

The process of labeling data for annotation is time-consuming. As a result, businesses find it difficult to determine how much money they need to create an ML & AI training project.

Businesses occasionally have to move backwards as a result of paying a large workforce a significant wage for a longer length of time and investing in expensive technologies.

Additionally, setting up a sizable, ergonomic office space with all the required amenities is difficult for businesses to handle.

failure to abide by data security rules-

Because of a severe lack of process understanding, data annotation organizations struggle to adhere to international data security standards. As Big Data becomes more and more widespread, the standards governing data privacy compliance are becoming stricter.

When it comes to raw data, it contains incredibly private information such as reading texts, identifying faces, etc. Therefore, tagging misinformation or any little mistakes can have huge repercussions. Besides, the data leak is the most important factor to be addressed here.

Hence, data labeling companies are sometimes failing to comply with these compliances with privacy and internal data security standards.

**Problem Statement:**

In the rapidly evolving landscape of machine learning applications for businesses, the crucial bottleneck lies in the accurate and efficient annotation of vast datasets. As organizations strive to develop robust machine learning algorithms to enhance decision-making processes, they encounter challenges in managing annotation projects. These challenges encompass diverse aspects such as the need for specialized tools, user interface (UI) design considerations, the integration of annotation processes into existing workflows, and the inevitable turnover of annotation personnel.

The problem at hand is to establish a seamless and scalable data annotation framework that aligns with business goals, accommodates various project requirements, integrates smoothly with machine learning algorithms, provides an intuitive user interface for annotators, employs efficient annotation tools, and addresses the issues arising from workforce turnover. Balancing the intricacies of these components is essential to ensure the consistent generation of high-quality annotated data sets, which in turn is pivotal for the success of machine learning initiatives within the organization.

.

Data Description:

CIFAR10 open source dataset will be used. It has 60000 color images with 10 classes, with 6000 images per class. 50000 training images and 10000 test images.The Dataset would divide training images into 5% training set ,5% validation set, 70% unlabelled data & rest 20% data is already clubbed as test data & then we will evaluate accuracy whenever required.

The dataset size is 177 MB. Here are the classes in the dataset

Airplane,automobile,bird,cat,deer,dog,frog,horse,ship,truck

Data type is a dictionary and there are 4 labels- dict\_keys([b'batch\_label', b'labels', b'data', b'filenames']), among these labels' & b data’ are important features. B Data' consists of an array shape of images & b labels’ contains true labels.

It is high dimensional with 3072 dimensions.

Pandas,Numpy,sklearn etc libraries will be used to process the data.

Labeled dataset is chosen in order to pass labels to the data whenever is required in order to help the model to enhance performance & to estimate the accuracy of the predicted labels by the model.

Tools (Pandas, SQL, Spark etc) that you will use to process this data Here, we will be using Python along with the below-listed libraries

Pandas,

NumPy,

matplotlib,

Scikit-learn

LIME

**Approach-Methodology**

Theoretical Approach

Label a small portion of data. Lets say Labeled data as Dl & unlabeled data as Du where Du>> Dl means Du is significantly larger than Dl.

Lets train the initial model M0 with Dl & use M0 to predict class labels. 3. From the unlabeled dataset Du,we‘ll smartly pick /sample a small subset of data s1 which could be hardly 1% of Du & obtain labels for this s1. When we smartly sample data, it means our target is to enhance the performance of our model M0.

Now, we’ll take our existing model M0 which is trained on previously labeled data Dl and we’ll add this newly labeled data s1, and we can either retrain the whole model from scratch or can retrain model M0 to obtain M1.

Repeat step 3 & step 4 until we obtain all labels correctly.

The stopping criteria would be when there is no significant improvement in model M1’s performance compared to M0.

You would have to determine the type of scenario your would like to use (that is, Membership Query Synthesis, Stream-Based Selective Sampling or Pool-Based sampling)

Query strategies

Algorithms for determining which data points should be labeled can be organized into a number of different categories, based upon their purpose-

Expected error reduction: label those points that would most reduce the model's generalization error.

Uncertainty sampling: Predict class probabilities for all unlabeled data using the trained classifier. If a classification has a probability higher than a predetermined threshold, it is assumed to be accurate. Include these samples in the training set along with their expected classes. label those points for which the current model is least certain as to what the correct output should be.

When a Supervised Machine Learning model makes a prediction, it often gives a confidence in that prediction. If the model is uncertain (low confidence), then human feedback can help.

Getting human feedback when a model is uncertain is a type of Active Learning known as Uncertainty Sampling.

Among types of Uncertainty Sampling,we can use

Least Confidence: difference between the most confident prediction and 100% confidence

OR

Entropy: difference between all predictions, as defined by information theory

Empirical/Experimental approach

We have not yet finalized on the approach of the experiment. Most probably, the sampling strategy we will use is a combination of uncertainty sampling and pool-based sampling.

And the model to be used is most likely to be an ensemble of CNNs along with Transfer learning at the top of everything. Transfer learning basically leverages the already built neural network architecture & initial dense layers are frozen as per our requirement.

The neural network in multi-class classification has the same number of output nodes as classes. Each output node produces a score for the class to which it belongs.

An activation layer is used to pass through scores from the previous layer. The score is

converted into probability values by the activation layer. The data is finally classified to the class with the highest probability value.Activation function could be Softmax or sigmoid activation function in the final layer, but we are yet to finalize it.

Each class's final score should be independent from the others. Most likely, sigmoid should be used because, unlike softmax, which converts each score of the final node between 0 and 1 independently of the previous scores, Sigmoid converts each score between 0 and 1 probabilities considering other’s score.

For Productionisation, We would use separate systems for model training(compute cluster ) & serving web application / web API.There would be data store (which is also storing feedback that users are giving , needs to be run periodically in regular intervals) and the model get retrained (either with entire combined dataset or only with recent data or only those incorrect labeled ones corrected by user as per the business needs)

Retrained model is pushed to Model registry/repository with its version.Its redeployed to the API.Use of pretrained weights from previous layer/model to new layer for new dataset and so on, & updating the model as new data arrives.

The majority of ML algorithms demand that any input or output variables be numbers. Any categorical data must therefore be converted to integers.

With one-hot, we create a new category column for each categorical value and give it a binary value of 1 or 0.

A binary vector is used to represent each integer value. The index is denoted by a 1 and all values are zero.

Dataset is balanced ,hence doesn't require any sampling . Moreover it requires image augmentation which we will be performing.

Furthermore, we could explore more for the dataset or can manipulate the balance of our CIFAR-10 dataset by techniques like upsampling,downsampling.

Key Performance metric (KPI) -

In the real world, there are many different types of metrics. We'll examine at a few of the mostly useful used KPIs that we might utilize in this multi class classification project-

Accuracy

Confidence score

**Interpretability-** We will be using LIME.

Model-Agnostic, Different Data Types (Tabular/Text)

Easy to interpret for Non-ML users (even non-Techies)

Relatively faster than SHAP in many case

5. Requirement Analysis

*Functional Requirements*

Purpose

To build a system which will predict the URL Data labels correctly, basically an automated image annotation system. The system will need retraining the model redundantly until a certain threshold accuracy is reached.

Inputs

A CSV File containing list of URL’s are provided as the input to the system.

Output

The system interprets & predicts the URL Data labels as Malicious and Benign.

Usability

It reduces manual efforts,saves time, higher accuracy,scalability,streamlined process, on time delivery, reduces cost.

*Software Requirements*

Notepad++ or any Code Editor

Anaconda Navigator , Jupyter Notebook, VS code & Streamlit.

Python 3

Python libraries like pandas, NumPy etc.

scikit-learn

Client environment may be Windows or Linux

Web Browser

6. Preprocessing - EDA and Feature Extraction

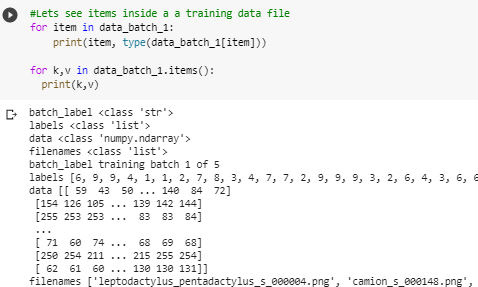
Dataset-

Link- <https://www.kaggle.com/datasets/siddharthkumar25/malicious-and-benign-urls?resource=download>

There are a total 7 files inside the dataset- 5 training, 1 test & one metadata file.

The dataset size is 177 MB. Here are the classes in the dataset -

[Airplane,automobile,bird,cat,deer,dog,frog,horse,ship,truck]



Data type is a dictionary and there are 4 labels- dict\_keys([b'batch\_label', b'labels', b'data', b'filenames']), among these labels' & b data’ are important features. B Data' consists of an array

shape of images & b labels’ contains true labels.

It is high dimensional with 3072 dimensions.

Pandas,Numpy,sklearn etc libraries will be used to process the data.

Labeled dataset is chosen in order to pass labels to the data whenever is required in order to help the model to enhance performance & to estimate the accuracy of the predicted labels by the model.

Loading the whole dataset-



Understanding how reshape & transpose works on single image-

The original batch data is a numpy array with the dimensions (10000 x 3072) of a dimensional tensor, where the column count (10000) represents the number of sample data. The row vector

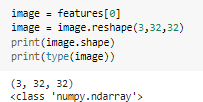
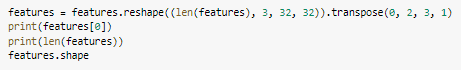
(3072), as indicated in the CIFAR-10/CIFAR-100 dataset, depicts a color image with a 32x32 pixel size. The row vector (3072), which will be used in this project to perform the classification tasks, is an inappropriate type of image data to feed. The dimensions of the tensor representing an image data should be either (num channel x width x height) or (num channel x width x height) in order to input the data into an NN model.

Our image data is a numpy ndarray.

Each image is a 1-D array having 3,072 entries. First 1024 entries for Red, then next 1024 are Green and lastly 1024 entries are Blue channels. In order to visualize the images we have to change the shape of the image as (32,32,3).

Reshape And Transpose a Single Image Our image is a single dimension array of size 3072. First 1024 entries of the array are of Red channel, next 1024 entries are of Green channel, and last 1024 entries are of Blue channel. Total 3072 entries are of three RGB channels.

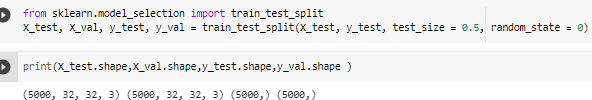
First we reshape the image/ array into (3,32,32) using image.reshape(3,32,32). (32,32) for 32x32 =1024 entries and 3 for RGB channels.





Creating Validation dataset

Lets split test dataset into 2 halves and use one half 5000 images as validation set



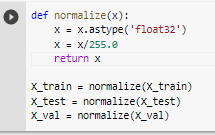
5. Normalize/Rescaling the dataset

The normalize function converts the input image data, x, into a normalized Numpy array. Without altering the array's shape, the original data's values will be changed to fall between 0 and 1, inclusive. An easy explanation for why normalization should be carried out has something to do with activation function.

As an illustration, the sigmoid activation function converts an input value to a new value between 0 and 1. The output value quickly reaches the maximum value 1 when the input value is reasonably large. Similar to this, when the input value is a little low, the output value easily reaches the maximum value of 0.

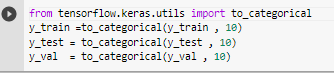
Pixel value of the image falls between 0 to 255.

So, we are scaling the value between 0 to 1 by dividing each value by 255



One Hot Encoding-

We would do one-hot encoding using tensorflow’s to\_categorical module-



The majority of ML algorithms demand that any input or output variables be numbers. Any categorical data must therefore be converted to integers.

With one-hot, we create a new category column for each categorical value and give it a binary value of 1 or 0.

A binary vector is used to represent each integer value. The index is denoted by a 1 and all values are zero.Given that the output of our model would provide the probability of the various categories in which a picture should be classified as a prediction. A vector with the same number of elements as the total number of image classes should exist. For instance, CIFAR-10 offers 10 different image classes, so we also require a vector with a 10 size. Each component represents the class-specific prediction probability.

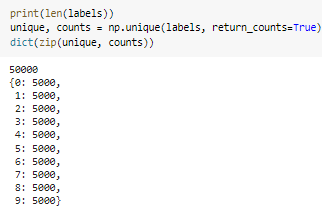
A comparison between the forecast and the ground truth label should also be possible using our model. It implies that the label data's shape should likewise be converted into a vector with a 10x10 size. Instead, we assign the value 1 to the relevant element because the label is the actual reality.

Even while this is useful in some ordinal circumstances, some input data lacks ranking for category values, which can cause problems with predictions and poor performance. One hot encoding comes to the rescue at that point.

Our training data is more useful and expressive thanks to one hot encoding, and it is also simple to scale. We can more quickly calculate a probability for our values by using numerical values.

For our output values in particular, one hot encoding is chosen since it offers more precise predictions than single labels.

This approach can result in a significant issue (too many predictors) if the original column contains a lot of unique values because it creates a lot of additional variables. One-hot encoding also has the drawback of increasing multicollinearity among the numerous variables, which reduces the model's accuracy.



Hence either , we could explore more for the dataset or can manipulate the balance of our CIFAR-10 dataset by techniques like upsampling,downsampling.

Secondly, we already have labels in order to assess our model’s accuracy, which is not the case with the real world projects,

f .ImageDataGenerator

ImageDataGenerator will be used to apply multiple data augmentation techniques to avoid overfitting in the training part.

from tensorflow.keras.preprocessing.image import ImageDataGenerator

datagen = ImageDataGenerator( rotation\_range=15, width\_shift\_range=0.1, height\_shift\_range=0.1,horizontal\_flip=True,)

datagen.fit(X\_train)

9. Visualization

After preprocessing & transforming dataset, Lets visualize images using matplotlib-



Model Building and training

There are 2 ways to program CNN with keras:

Sequential approach: Here, we generally add layers in sequence.

Modular approach: This is more important. This is more dynamic, customized, molded and easy to explore.

Effectiveness in both approaches will remain the same.

We would be choosing CNN keras sequential model as baseline model which is a deep learning model

Steps of Deep learning Analysis- Select Dataset

Pre-process Data Model Generation Model Compilation Model Training Model Evaluation Model Prediction

Build Baseline model-

Here building custom baseline model consisting of 5 blocks- 3 Conv2D layers( Feature Detection / extraction Block),

1 Flatten (Transition Block (from feature detection to classification)) , 4 Dense layers (Classification Block)

Below parameters are added as different layers in the model in order to tune the model to avoid overfitting and tune the model-

Batch normalization Dropout

Flatten

# Modelling - Model on CNN import tensorflow as tf import os

from keras.utils.vis\_utils import plot\_model from tensorflow.keras import models, layers

from keras.layers import Conv2D, MaxPooling2D, Flatten , Dense,

Activation,Dropout,BatchNormalization

# create a sequential model i.e. empty neural network which has no layers in it. model=tf.keras.Sequential([

#==================== Feature Detection / extraction Block ====================# # Add first convolutional block we use Conv2D and for colour images and shape use Conv3D

#Here filters = 64, kernel\_size= 3\*3 , activation = ‘relu’ and L2 regularizer used to avoid overfitting Conv2D(64,(3,3),input\_shape=(32,32,3),activation='relu',padding='same',kernel\_reg ularizer=tf.keras.regularizers.l2(0.01)),

BatchNormalization(), Conv2D(64,(3,3),input\_shape=(32,32,3),activation='relu',padding='same',kernel\_reg ularizer=tf.keras.regularizers.l2(0.01)),

BatchNormalization(), MaxPooling2D(pool\_size=(2,2)), Dropout(0.25),

Dropout(0.25),

# Add Second convolutional block Conv2D(128,(3,3),activation='relu',padding='same',kernel\_regularizer=tf.keras.reg ularizers.l2(0.01)),

BatchNormalization(), Conv2D(128,(3,3),activation='relu',padding='same',kernel\_regularizer=tf.keras.reg ularizers.l2(0.01)),

BatchNormalization(), MaxPooling2D(pool\_size=(2,2)), Dropout(0.25),

Dropout(0.25),

# Add Third convolutional block Conv2D(256,(3,3),activation='relu',padding='same',kernel\_regularizer=tf.keras.reg ularizers.l2(0.01)),

BatchNormalization(), MaxPooling2D(pool\_size=(2,2)), Dropout(0.25),

Dropout(0.25),

#==================== Transition Block (from feature detection to classification)

====================#

# Add Flatten layer. Flatten simply converts matrics to array Flatten(input\_shape=(32,32)),

#==================== Classification Block ====================#

# Classification segment - fully connected network

# The Dense layer does classification and is a deep neural network. Dense layer always accept the array.

Dense(128, activation='relu'), Dense(100, activation='relu'), Dense(80, activation='relu'),

# Add the output layer

model.add(layers.Dense(10, activation='softmax')) # as Output layer in above image. The output layer normally have softmax activation

])

Compile model-

A model's behavior is "frozen" when compile() is called on it. The values of the trainable attributes at the time the model is compiled should, according to this, be maintained throughout the model's existence until compile is called again.

#Compile defines the loss function, the optimizer and the metrics. model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])

#The summary is textual and includes information about:

#The layers and their order in the model. The output shape of each layer. The number of parameters (weights) in each layer. The total number of parameters (weights) in the model.

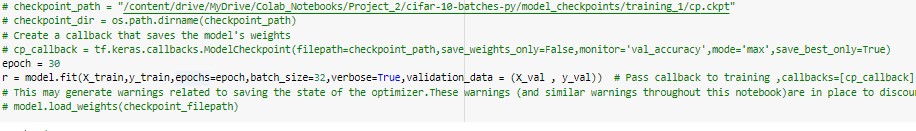
model.summary()

Parameters summary- Total params: 1,104,094

Trainable params: 1,102,814

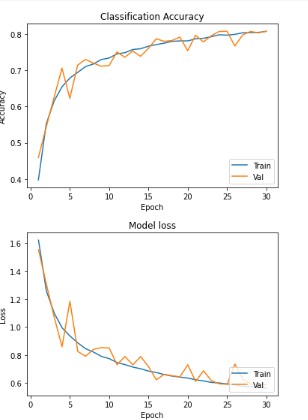
Non-trainable params: 1,280

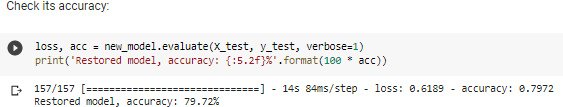
Training model-



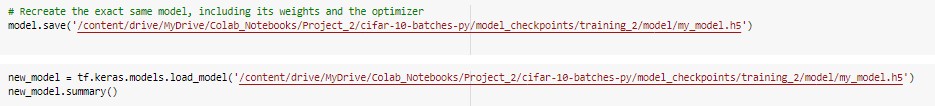
Evaluating model on Test data-

Accuracy is approx 80% on test dataset , also there is no gap in graph between accuracy & loss of training & validation datasets.

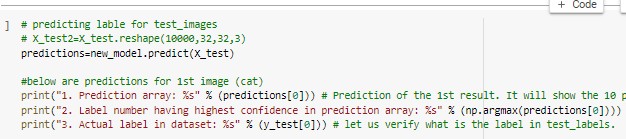




Save & Load model

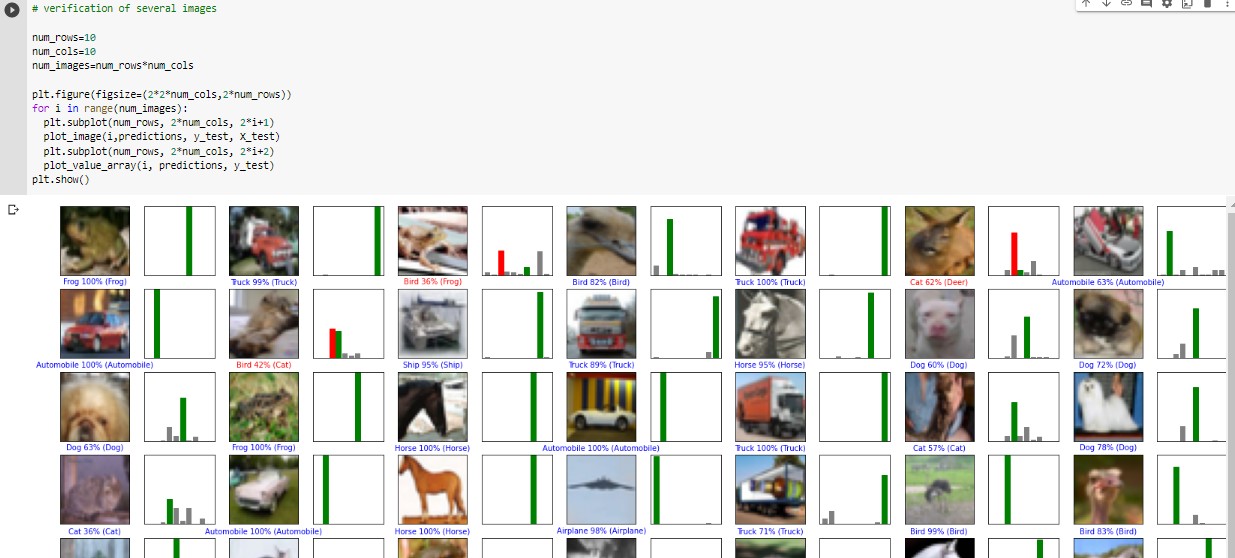


Testing the model & make prediction on test data

model.predit() function returns confidence score array for an image for all 10 labels, y\_pedicted will be the max value in predictions array.

Visualizing Test data prediction

Plot represents the image, predicted label ,(confidence score probability in percentage), ground truth and histogram of prediction array

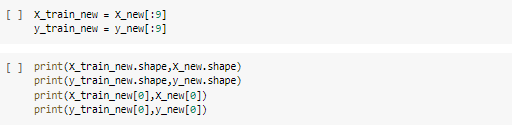


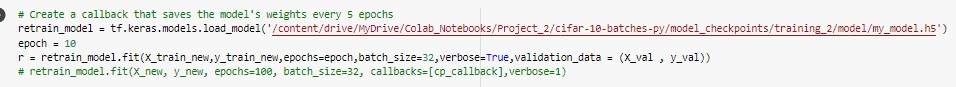
Model retraining -

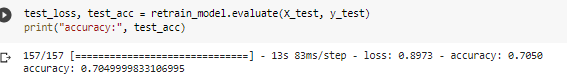
Retrain the model to have them fitted on wrong predicted images & save the retrained model.

We will keep the validation set (along with their ground truth values) same throughout the process & will check whether model’s performance on test dataset is improving or not ,

If yes, then we will deploy this new model , otherwise the existing model would be continued.







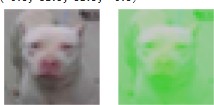
Observation-

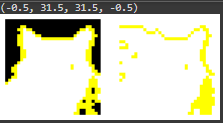
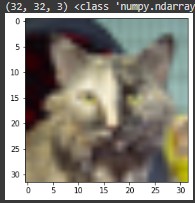
Model's accuracy goes down, hence it can't be considered.

10. Interpretability - Explaining Model predictions

LIME explanation is used here.







Observations-

For First 2 images, the greener & darker part are reflecting correctly ,however its expected to be more precise in pointing boundaries

For next 2 incorrect predictions, The explanations are highlighting correct areas in green to be considered in predictions , however ,the predictions are still wrong..

LIME applied on images other than CIFAR 10 dataset projects super pixels rather than depicting green/ red regions.

We would be trying out other approaches to retrain models & improve accuracy.

Model Improvement Strategy-

Now here comes the main part, in order to improve the model’s performance, the technique which we would be implementing is active learning & transfer learning which consist of few steps-

loading base model, add few more layers considering it will not harm the model,

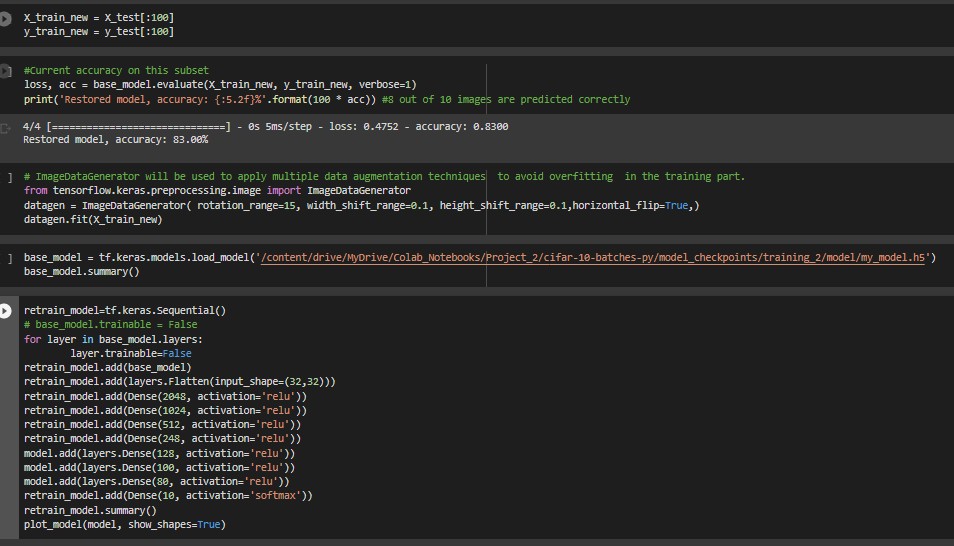
retraining model to learn weights from new data only ( which includes incorrect predictions). In addition to that, Ensure to keep the initial training weights from training data the same.

If the model performance gets improved , we will consider it as our new model to be deployed in production , otherwise the existing model will be retained.

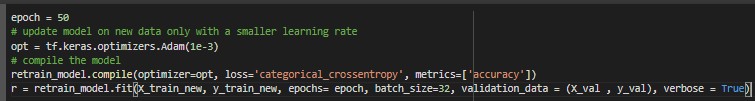
a. Active Learning and Transfer Learning

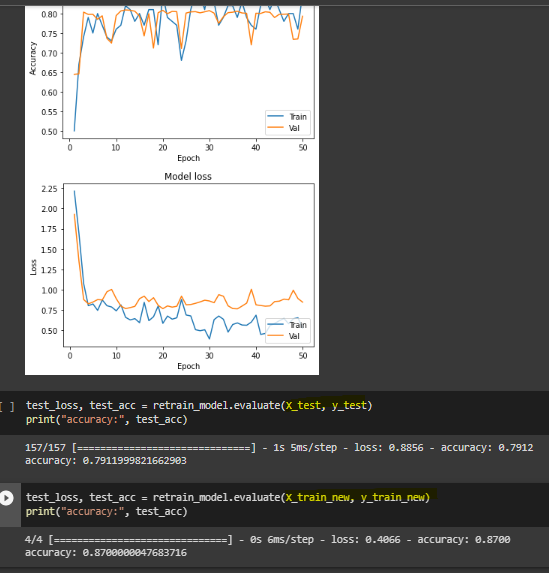
The initial baseline model is restructured by adding a few more dense layers in it.

We will keep its initially trained weights untouched by setting layer.trainable = False



This new model will be trained only on new weights on user supplied test data to have it fitted on wrong predicted images.

The model is trained upto 30 epochs , alternatively, it can be trained with early stopping keeping a certain patience threshold.



b.Observation-

The results are good, performance has improved on previously incorrectly predicted images and overall accuracy is the same.

Additionally, a round of fine-tuning of the entire model to give a final touch. unfreeze the base model and train the entire model end-to-end with a very low

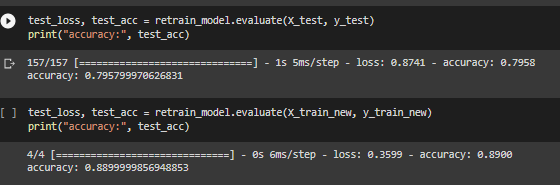
learning rate.

base\_model.trainable = True retrain\_model.summary()

retrain\_model.compile( optimizer=tf.keras.optimizers.Adam(1e-5), # Low learning rate loss='categorical\_crossentropy', metrics=['accuracy'])

epoch = 15

r = retrain\_model.fit(X\_train\_new , y\_train\_new , batch\_size = 32, epochs = epoch , validation\_data = (X\_val , y\_val), verbose = True)

If it enhances the performance a bit, then we can save it as the final model , otherwise it can be ignored,

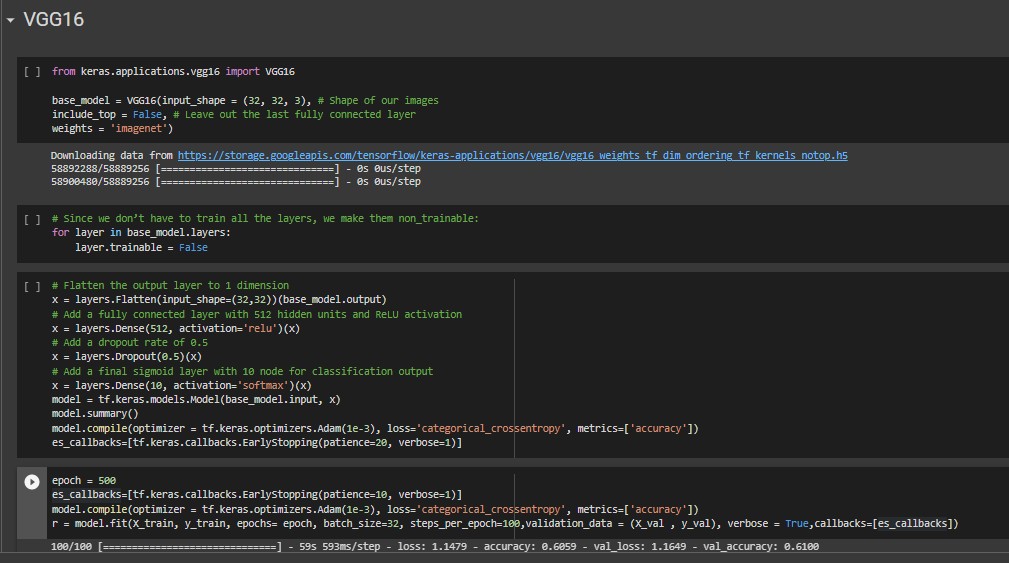
Observation-

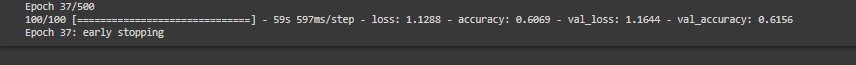
In our case, it helps in reaching accuracy better than the previous one.

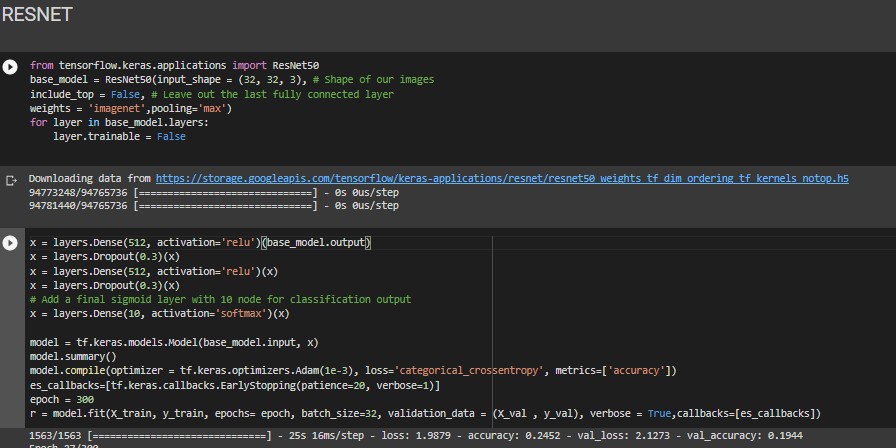
Getting good results on using transfer learning on existing model & later doing finetuning. Accuracy improvement is directly proportional to the amount of data used in transfer learning.

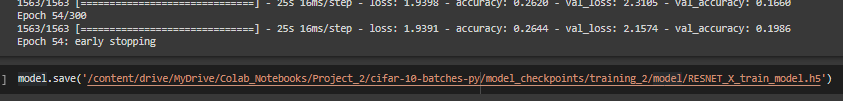
Training keras other pretrained models

We tried out our test dataset with VGG16 & RESNET models.









f.Observations-

VGG16 & RESNET models are giving approximately 50% and 20% accuracy respectively .

Pretrained models are under performing on the test data when compared to our custom model ,

**Conclusion-**

We will go ahead with our custom model & deploy it in production.

Deployment and Productionisation

The final phase of this project.is Deployment, here we are deploying the whole machine learning pipeline into a production system, into a real-time scenario.

In this final stage we need to deploy this machine learning pipeline to put of research available to end users for use. The model will be deployed in real world scenario where it takes continuous raw input from real world and predict the output.

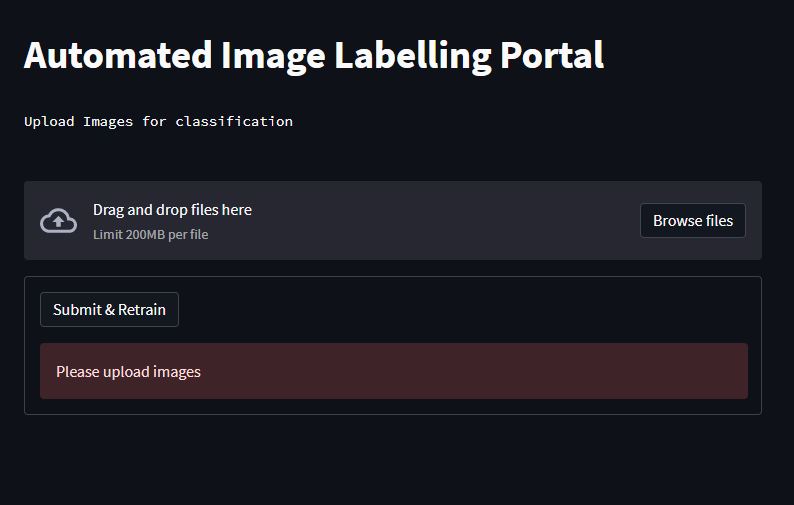
App Building:

In this step a web application developed using Streamlit. It provides a functionality to the user for uploading the input in csv format.

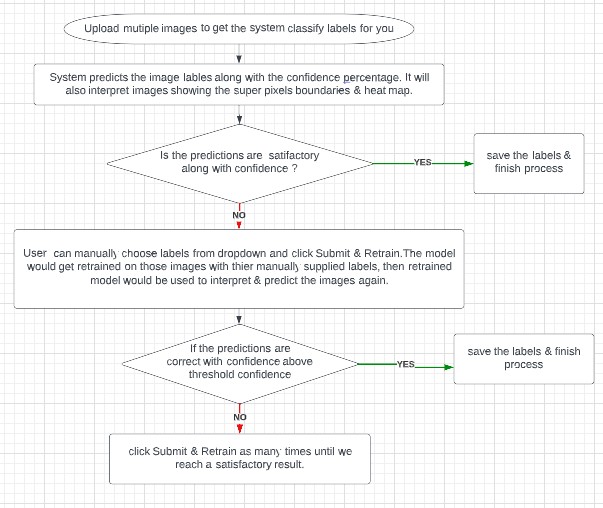
User interface

User interface is built on a Streamlit application which is an open source platform.

The code is written in stream\_app.py file which will use baseline model & retrained model to showcase how active learning improves prediction accuracy & confidence scores on the predicted labels.

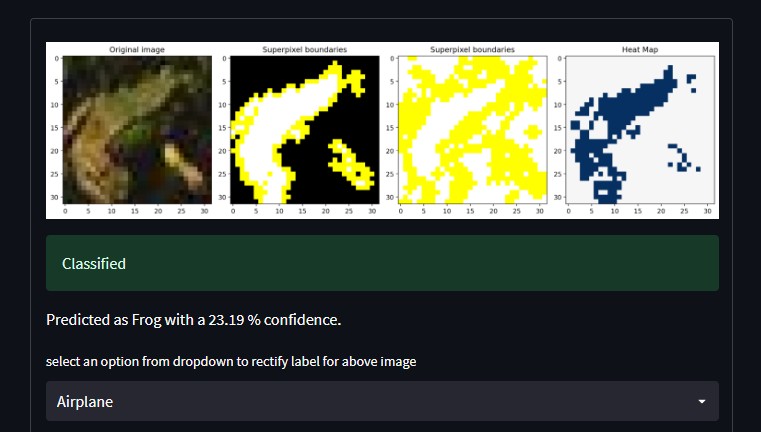


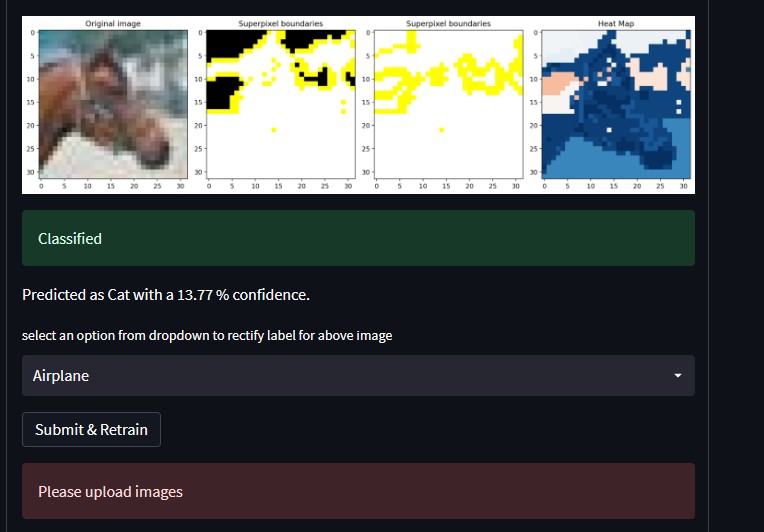
Flowchart



Working

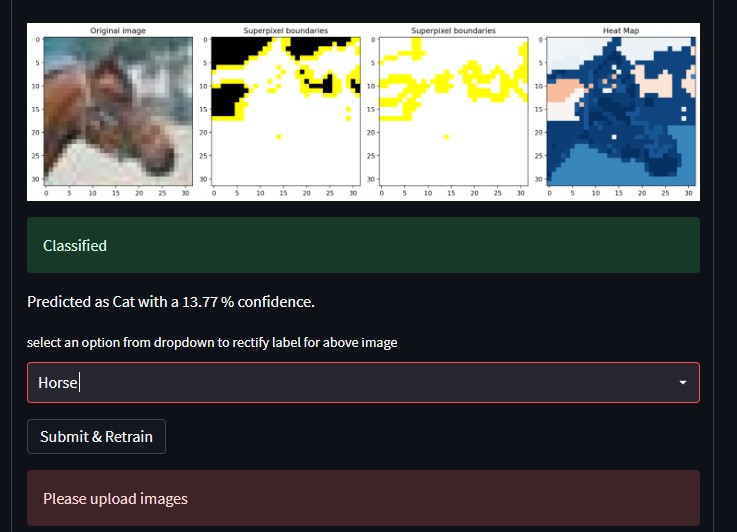
Once images are uploaded in order to classify as one of the 10 labels, System predicts the image labels along with the confidence percentage. It will also interpret images showing the super pixels boundaries (positive & negative ) & heat map plot. consisting of blue & red areas.

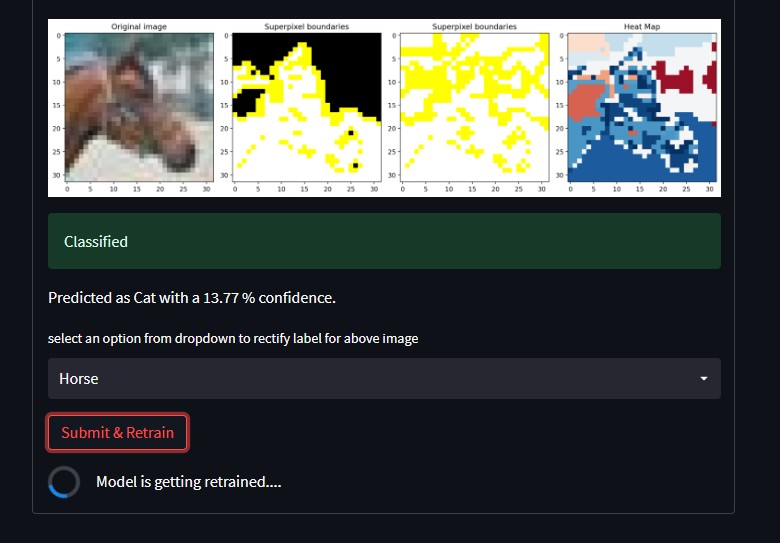


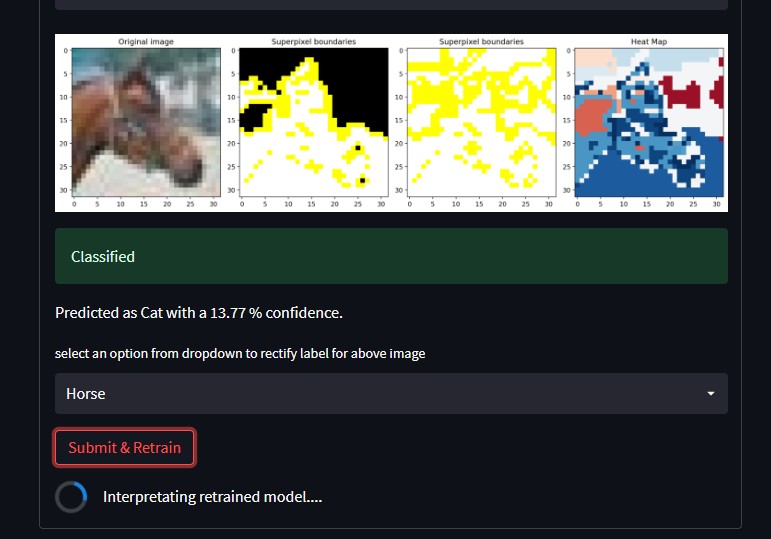


Whether the predictions are satisfactory along with confidence, then, we can save correctly satisfied labels along with images & can have our unlabelled dataset transformed to a labeled one.

User can manually choose labels from dropdown and click Submit & Retrain.The model would get retrained on those images with their manually supplied labels, then the retrained model would be used to interpret & predict the images again.

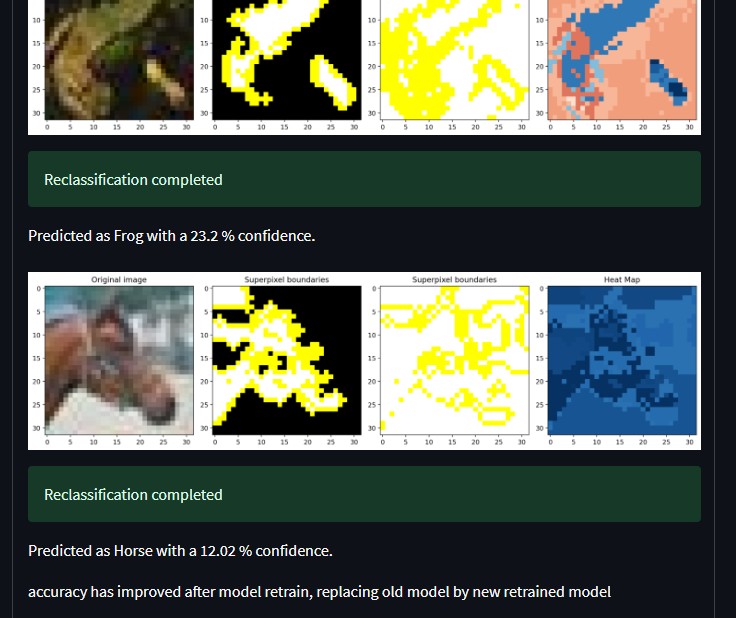






If the predictions are correct with confidence above threshold confidence

save the labels & finish process.Otherwise, click Submit & Retrain as many times until we reach a satisfactory result.

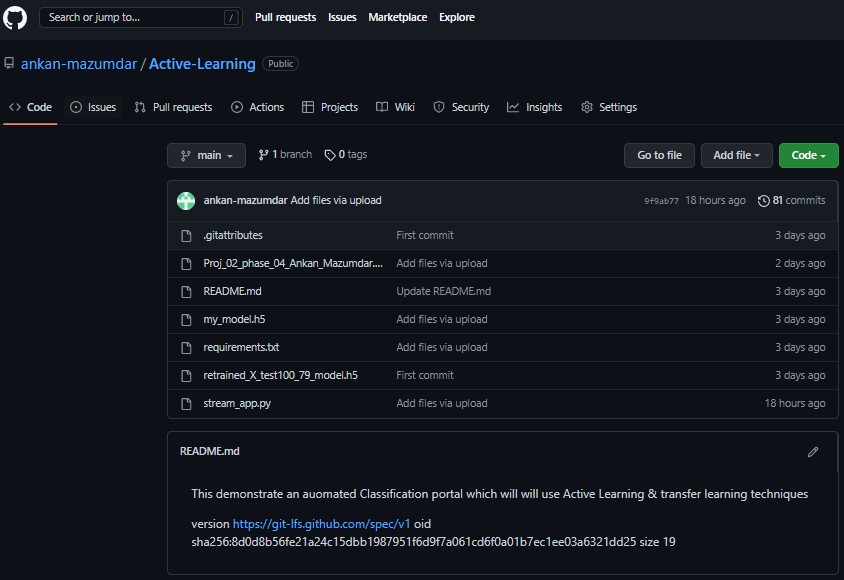


Productionisation-

App is Deployed to streamlit cloud, Please access below link-

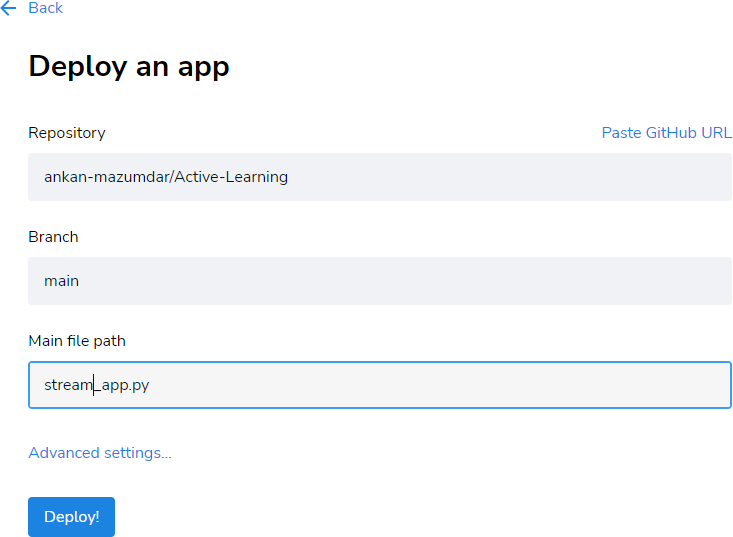
<https://ankan-mazumdar-active-learning-stream-app-kbbcqv.streamlitapp.com/>

Steps to deploy-

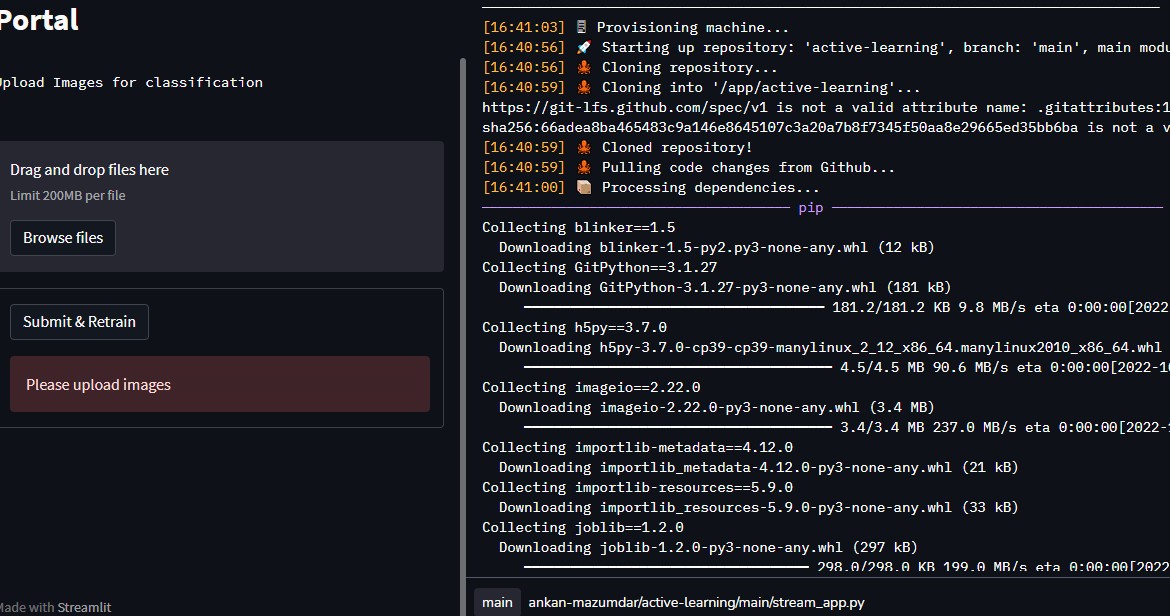
Upload file from local to github repo- https://github.com/ankan-mazumdar/Active-Learning

For uploading large files in git ,have followed the git lfs documentation- <https://git-lfs.github.com/> Add a requirements file for Python dependencies.

Next, need to go to streamlit cloud deployment link- <https://share.streamlit.io/deploy> and enter the repo details & submit



Once deployment is done, The app will be loaded, the execution logs are loaded side by side which helps to verify if deployment completed successfully or to look for any errors.



Observations-

Streamlit provides an easy ,quick & beautiful deployment platform, however it has its own limitation of further customization.

Dependency on github as code repo, makes it difficult to create & save models in github in runtime.

Streamlit is consistent with no outages , however , hasn't tested its scalability nor how many users it can handle at the same time.

Model is trained only on the CIFAR 10 dataset. API is limited to predict only 10 classes present in the dataset itself, any object intended to be classified out of the 10 classes in CIFAR 10 would be invalid .

References-

<https://python-data-science.readthedocs.io/en/latest/activelearning.html>

<https://paperswithcode.com/task/active-learning/latest>

https://www.diva-portal.org/smash/get/diva2:1586990/FULLTEXT01.pdf

[https://www.datacamp.com/tutorial/active-learning](http://www.datacamp.com/tutorial/active-learning)

<https://en.wikipedia.org/wiki/Active_learning_(machine_learning)>

https://drive.google.com/file/d/11Rqk0tGHExrb2LwoecUwPSPr0Y6VE-he/view

[https://towardsdatascience.com/identifying-the-right-classification-metric-for-yo](https://towardsdatascience.com/identifying-the-right-classification-metric-for-yourtask-21727fa218a2) [urtask-21727fa218a2](https://towardsdatascience.com/identifying-the-right-classification-metric-for-yourtask-21727fa218a2)

[https://www.kaggle.com/code/nkitgupta/evaluation-metrics-for-multi-class-classi](http://www.kaggle.com/code/nkitgupta/evaluation-metrics-for-multi-class-classi) fic ation/notebook

https://keras.io/api/optimizers/

<https://www.tensorflow.org/api_docs/python/tf/keras/Sequential>

[tensorflow conv2d](https://www.tensorflow.org/api_docs/python/tf/nn/conv2d)

[Batch Normalization (the original paper)](https://arxiv.org/abs/1502.03167)

[Dropout (the original paper)](https://www.cs.toronto.edu/~hinton/absps/JMLRdropout.pdf)

[Understanding Dropout / deeplearning.ai — Andrew Ng.](https://www.youtube.com/watch?v=ARq74QuavAo)

[Tensorflow Softmax Cross Entropy with Logits](https://www.tensorflow.org/api_docs/python/tf/nn/softmax_cross_entropy_with_logits_v2)

[Tensorflow Reduce Mean](https://www.tensorflow.org/api_docs/python/tf/reduce_mean)

[Tensorflow Optimizers](https://www.tensorflow.org/api_guides/python/train)

[An overview of gradient descent optimization algorithms](http://ruder.io/optimizing-gradient-descent/)

[Optimization for Training Deep Models](http://www.deeplearningbook.org/contents/optimization.html)

[Tensorflow tf.train.AdamOptimizer](https://www.tensorflow.org/api_docs/python/tf/train/AdamOptimizer)

<https://www.youtube.com/watch?v=HxtBIwfy0kM&t=105s>

<https://www.kaggle.com/code/muhammadhafil/image-classification-basics>

<http://yann.lecun.com/exdb/publis/pdf/lecun-99.pdf>

●

https://numpy.org/doc/stable/reference/generated/numpy.transpose.html

[https://www.tensorflow.org/api\_docs/python/tf/keras/preprocessing/image/Imag](https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator) [eDataGenerator](https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator)

https://matplotlib.org/stable/tutorials/introductory/images.html

<https://www.tensorflow.org/api_docs/python/tf/keras/utils/to_categorical>

https://medium.com/analytics-vidhya/a-tip-a-day-python-tip-8-why-should-we-n ormalize-image-pixel-values-or-divide-by-255-4608ac5cd26a

●

●

<https://docs.streamlit.io/>

[https://docs.github.com/en/billing/managing-billing-for-git-large-file-storage/view](https://docs.github.com/en/billing/managing-billing-for-git-large-file-storage/viewing-your-git-large-file-storage-usage) [ing-your-git-large-file-storage-usage](https://docs.github.com/en/billing/managing-billing-for-git-large-file-storage/viewing-your-git-large-file-storage-usage)

<https://www.geeksforgeeks.org/git-lfs-large-file-storage/>

<https://discuss.streamlit.io/t/oserror-unable-to-open-file-from-githib/29480/6>

https://git-lfs.github.com/

<https://share.streamlit.io/deploy>

https://keras.io/api/optimizers/

<https://keras.io/api/models/sequential/>

https://keras.io/guides/sequential\_model

<https://www.tensorflow.org/api_docs/python/tf/keras/Sequential>

[Tensorflow Reduce Mean](https://www.tensorflow.org/api_docs/python/tf/reduce_mean)

[Tensorflow Optimizers](https://www.tensorflow.org/api_guides/python/train)

[An overview of gradient descent optimization algorithms](http://ruder.io/optimizing-gradient-descent/)

[Optimization for Training Deep Models](http://www.deeplearningbook.org/contents/optimization.html)

[Tensorflow tf.train.AdamOptimizer](https://www.tensorflow.org/api_docs/python/tf/train/AdamOptimizer)

<https://www.youtube.com/watch?v=HxtBIwfy0kM&t=105s>

<https://www.kaggle.com/code/muhammadhafil/image-classification-basics>

<http://yann.lecun.com/exdb/publis/pdf/lecun-99.pdf>