IJNRD.ORG ISSN : 2456-4184
INTERNATIONAL JOURNAL OF NOVEL RESEARCH AND DEVELOPMENT (IJNRD) | IJNRD.ORG
An International Open Access, Peer-reviewed, Refereed Journal

DETECTION OF DIABETIC RETINOPATHY USING DENSENET

[1]Mrs. S. A. Bhavani, [2]L. Sai Revanth, [3]S. Sri Harini, [4]A. Yogindra, [5]V. Bharadwaj

[1]Assistant Professor,
[1]Computer Science and Engineering,
[1]Anits College, Visakhapatnam, India

ABSTRACT : DR and other microvascular and macrovascular disorders of diabetes cause serious illness and death and have a significant economic impact. Retinopathy and other minor microvascular disorders are caused by chronic hyper-glycemia, vascular injury and leakage, edema, tightening of the capillary basement membrane, neovascularization, bleeding, and ischemia.

In this paper, the technology in the previous decade have been increasing at a tremendous pace especially in fields like medicine, military etc. The process comprises two main phases i.e. transfer the learning of the process and its localization and its classification. The launched task consists of two main components, the first being data set and the other is an advanced Dense-Net network specialized in eye disease detection. Following this, Dense-Net is trained to distinguish existing classification on the degree of Diabetic Retinopathy. Finally, accuracy is measured in all units as per the metrics used in the field of computer vision classification.

Various training and testing of data takes place and test results show that the proposed model detects all DR categories and works better compared to the standard ones in the same Kaggle database. Experimenting with Densenet-121 we got an accuracy of 97.3%, sensitivity of 97.6%, precision of 97.7% and f1 score of 97.6%. This shows that it is very capable of accurately detecting level of Diabetic Retinopathy. Our finding will help in detecting early stages of DR to tackle further damage in eye.

Keywords - Diabetic Retinopathy, Densenet, Convolutional Neural Networks, Gaussian Blur, Convolutional Layer, Pooling.

INTRODUCTION

DR causes serious illness and death and have a significant economic impact. The prevalence of DR in India is 16.9%: Research while the detection rate of DR threatens is 3.6 percent. AI and Deep learning(DL) find applications in ophthalmology as a major part of the data to be analyzed based on images and the results are related to image recognition. recommendations. The role of AI in detecting DR (RDR) targeted, defined as retinopathy for DR (NPDR) and beyond, appears to promise high sensitivity and certain levels in many studies.

NEED OF THE STUDY

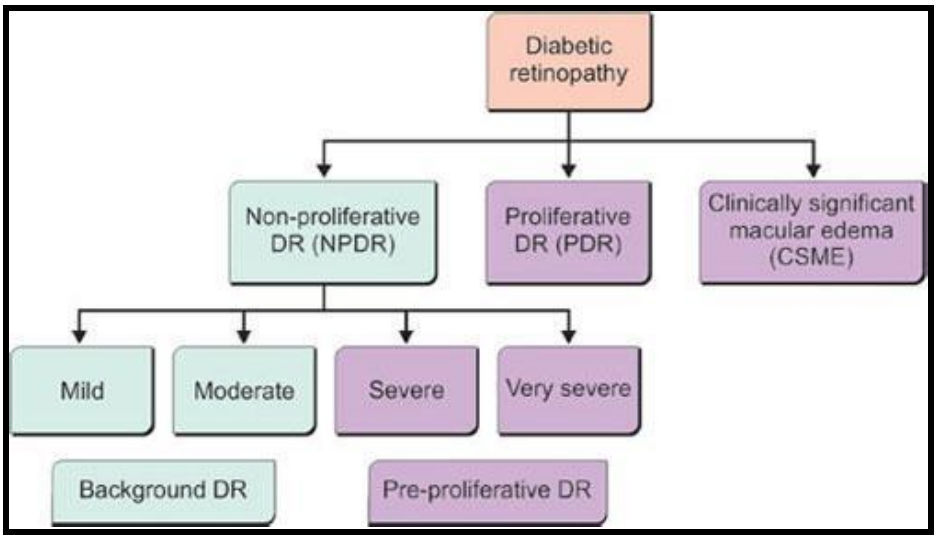
This project aims at developing a model for detecting diabetic eye diseases from a given set of retinal images using modern techniques such as deep learning. The main objective of this project is to use the deep learning capabilities in the medical field including ophthalmology. In-Depth learning and CNN are able to identify, localize and quantify the pathological and retinal characteristics and their effectiveness continues to grow. Our main purpose is to make an automated system using DENSENET, a deep learning approach that requires less hardware to detect DR and its level with optimal accuracy which is comparable to the industry-leading methods and time as compared to existing methods.

PROBLEM STATEMENT

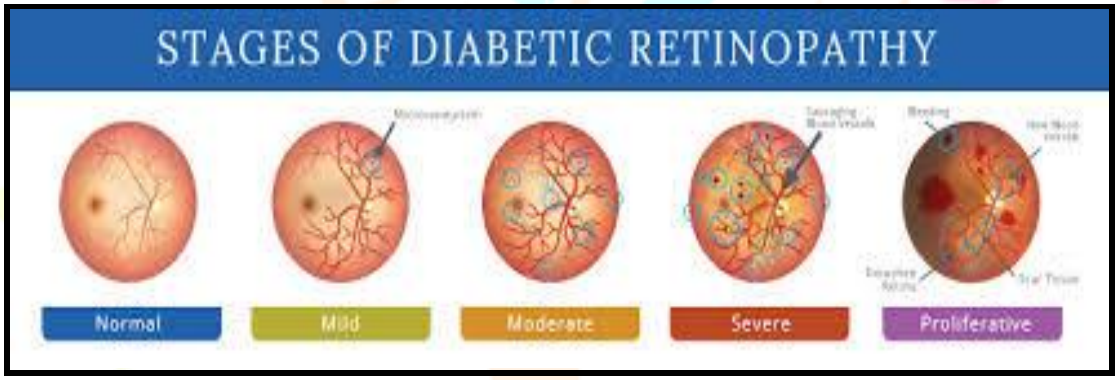
This project aims at developing a model for detecting diabetic eye diseases from a given set of retinal images using modern techniques such as deep learning. The main goal of this project is to use the deep learning capabilities in the medical field including ophthalmology. In-Depth learning and CNN are able to identify, localize and quantify the pathological and retinal characteristics and their effectiveness continues to grow. Our main purpose is to make an automated system using DENSENET, a deep learning approach that requires less hardware to detect DR and its level with optimal accuracy which is comparable to the industry-leading methods and time as compared to existing methods.

PROPOSED SYSTEM

Our goal is to provide a framework for Diabetic Retinopathy classification in which fundus images from various sources and lighting are used to detect its degree of DR. In this project we are using DenseNet-121, a type of Convolution Neural Network for our purpose. Aptos 2019 blind detection dataset and Diabetic Retinopathy Resized Dataset are used. The fundus images are taken preprocessed and classified into five types of DR. The five types are:



- Normal
- Mild
- Moderate
- Severe
- Proliferative



Our main purpose is to make an automated system using DENSENET, a deep learning approach which requires less hardware to detect DR and its level with optimal accuracy which is comparable to the industry leading methods and time as compared to existing methods. As the images in the dataset are not uniform and were taken in different lighting conditions, different size and magnification, and different orientation, it is challenging to make a model which is versatile which will work on the above factors to get high accuracy which can be compared to existing methods.

TECHNOLOGIES USED

- **Python:**

Python is a well-known programming language for computers that is used to build websites and applications, automate procedures, and analyse data. We use the Numpy and pandas libraries for our project.

- **Keras:**

To implement neural networks, Google created the high-level Keras deep learning API. It is created in Python and is designed to simplify the implementation of neural networks.Keras is fairly easy to learn and use since it provides a high level of abstraction Python frontend and the option of multiple back-ends for computation. As a result, Keras is slower than other deep learning frameworks but far more user-friendly for beginners. Using Keras, you may switch between different back ends. The following frameworks are supported by Keras:

  - Plaid ML
  - Tensorflow
  - Theano
  - MXnet
  - CNTK(Microsoft Cognitive Toolkit)

- **SKlearn:**

Skearn is the name of the most efficient and dependable Python machine learning package (Skit-Learn).With a consistent Python interface, it offers a variety of effective methods for statistical modelling and machine learning, including dimensionality reduction, clustering, and classification. This library, which is based on NumPy, SciPy, and Matplotlib, was mostly developed in Python.

- **TensorFlow:**

TensorFlow is an open-source library developed by Google especially for deep learning applications. Also supported is conventional machine learning. TensorFlow was first developed to perform massive numerical computations without having deep learning in mind. Multi-dimensional arrays are very useful when working with large amounts of data.

STAKEHOLDERS:

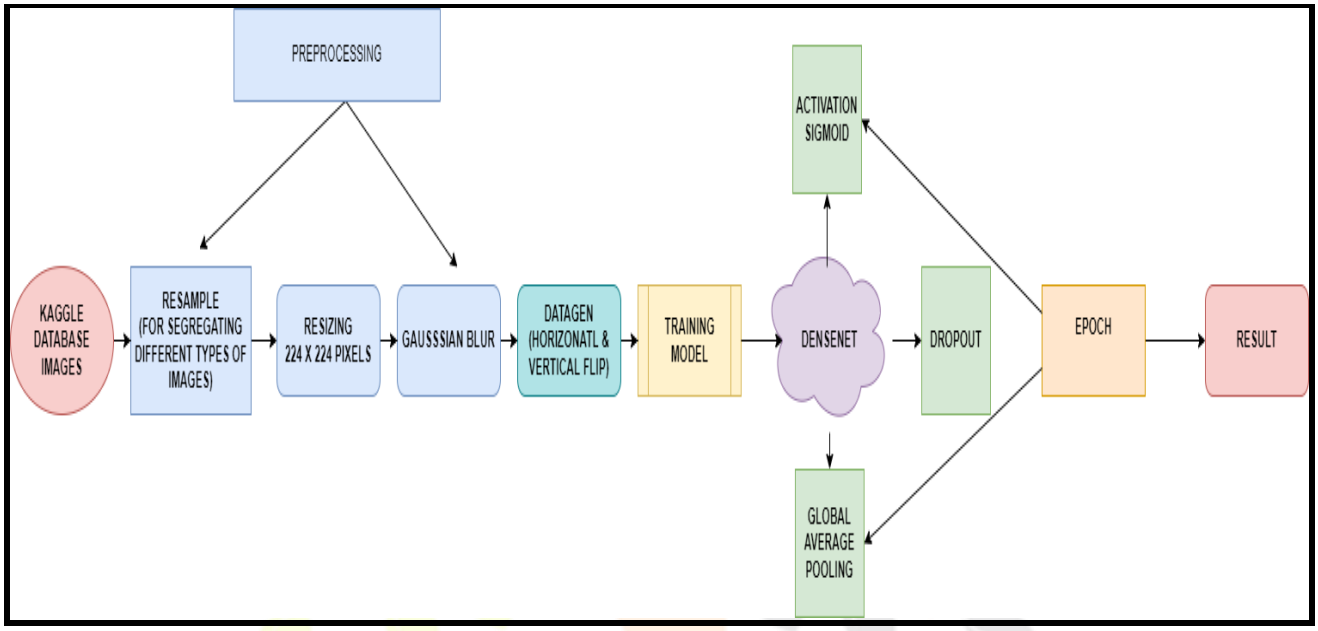
Anybody whose interests are impacted by the success or failure of a project or business venture is considered a stakeholder. The following applications for the suggested system are possible:

- Physicians
- Pharmacists
- Health Officials
- Health Management Experts
- Patient Representatives

PROPOSED METHODOLOGY:

In this article, we suggest a technique using Densenet-121.To implement this method steps shown in block diagram are followed.
First, the algorithm for image resampling, resizing and preprocessing is presented.
Second, data generator is created to for data augmentation to increase the number of datas during training
Finally, the data is fitted into our Densenet model to detect the severity and the level of Diabetic Retinopathy.

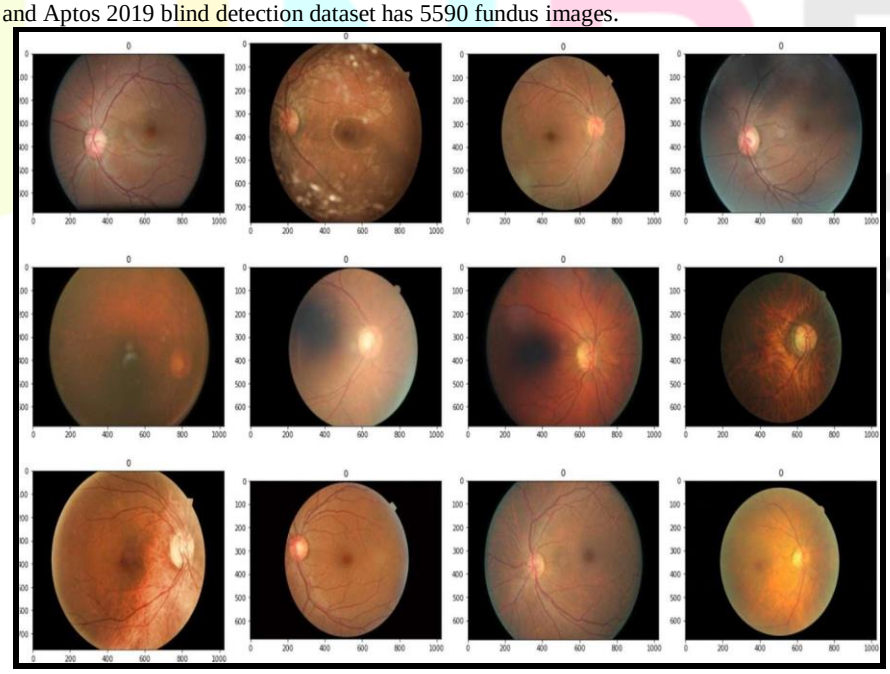
SYSTEM ARCHITECTURE:



The above block diagram was used for the implementation of the proposed method. First dataset consists of fundus image which is pre-processed by doing sampling, resizing, applying gaussian blur. Making data generator which helps in horizontal and vertical flips. Then data is fitted in in Densenet-121 model to get the result.

COLLECTING DATA:

We used Aptos 2019 blind detection dataset and Diabetic retinopathy resized dataset both publicly available in Kaggle website. The fundus images were taken in different conditions with different magnification and orientation. Diabetic retinopathy resized has 35216 fundus images and Aptos 2019 blind detection dataset has 5590 fundus images.



Diabetic Retinopathy Resized Dataset

DR Severity	Number of Images
Normal	25810
Mild	5292
Moderate	2443
Severe	873
Proliferative	708

Aptos 2019 Blind Detection Dataset

DR severity	Number of Images
Normal	1805
Mild	999
Moderate	370
Severe	295
Proliferative	193

PRE-PROCESSING

1. Resampling:

The dataset had around 35126 images of various types but was not uniform which could hamper our result so the images are resampled and final count of images droped to 9600.

Algorithm

```
if level <3
        sample(1200*2)
else
        sample(1200*1)
```

Taking 2400 count of level 0,1,2 DR and 1200 of type 3 and 4.

2. Resizing:

The photos are then downsized to 224x224, and the data is then stored in a single numpy array.Ratio is calculated first;

ratio = Height(new)/height= Width(new)/width (1)
Height(new) = ratio*height (2)
Width(new) = ratio*width (3)

Lets say; Images are of size 600px*600px:

ratio = 224/600 = 0.373333
Height = 0.373333*600 = 224px
Width = 0.373333*600 = 224px

3. **Image Filtering**

A photograph is blurred (smoothed) using a Gaussian function to provide the Gaussian blur feature in order to reduce noise. It can be compared to a non-uniform low-pass filter that reduces visual noise and unimportant details while preserving low spatial frequency. To do this, a picture is often convolved with a Gaussian kernel. This Gaussian kernel is represented in 2-D form as

$$G2Dxy\sigma=12\pi\sigma2e-x2+y22\sigma2$$

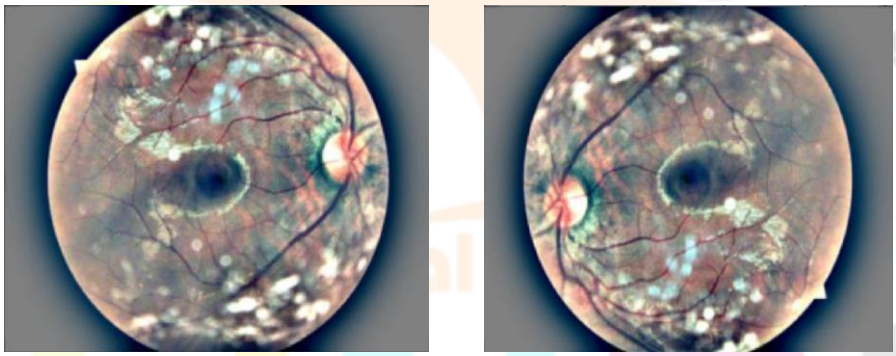
where the location indices x and y are given, and is the distribution's standard deviation.The value of determines the variance of the Gaussian distribution, which creates the extent of the blurring effect surrounding a pixel. We experimented with sigma values ranging from 0.1 to 16, and found that as sigma increases, the amount of high-frequency information surrounding a pixel decreases. In our research, we find that a standard deviation of 3 produces the best segmentation outcomes.

4. Data Augmentation:

With data manipulation, such as flipping data horizontally and vertically and randomly zooming in on photos, more data is available for training. Table 3 describes the data augmentation technique applied to datasets.

Table: Features used for data augmenting.

Feature	**Description**
zoom_range = 0.15	Configure the zoom range to be random.
horizontal_flip = True	The input image is mirrored along the y axis.
vertical_flip = True	The input image is mirrored along the x axis.
fill mode = constant	Filling points outside the input bounds is done with this.

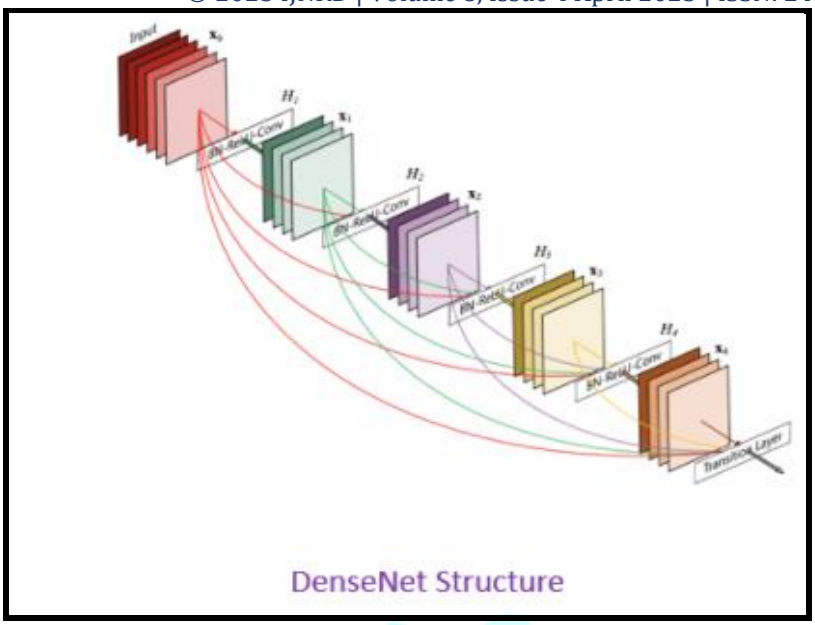


(a) (b)

Figure : Image after data generator (a)Horizontal (b)Vertical

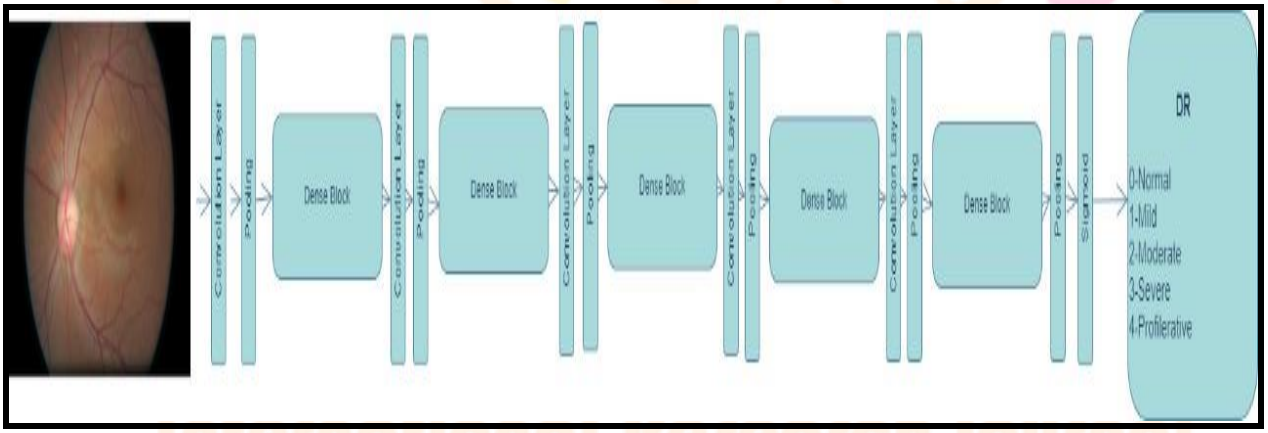
DENSENET-121:

- The data is fitted into the Densenet-121 model.
- By using the composite function, the current layer receives its input from the output of the layers that came before it.
- This composite procedure consists of the convolution layer, the pooling layer, the batch normalisation layer, and the non-linear activation layer.
- These functions in blocks together known as Dense Block.



Densenet Structure

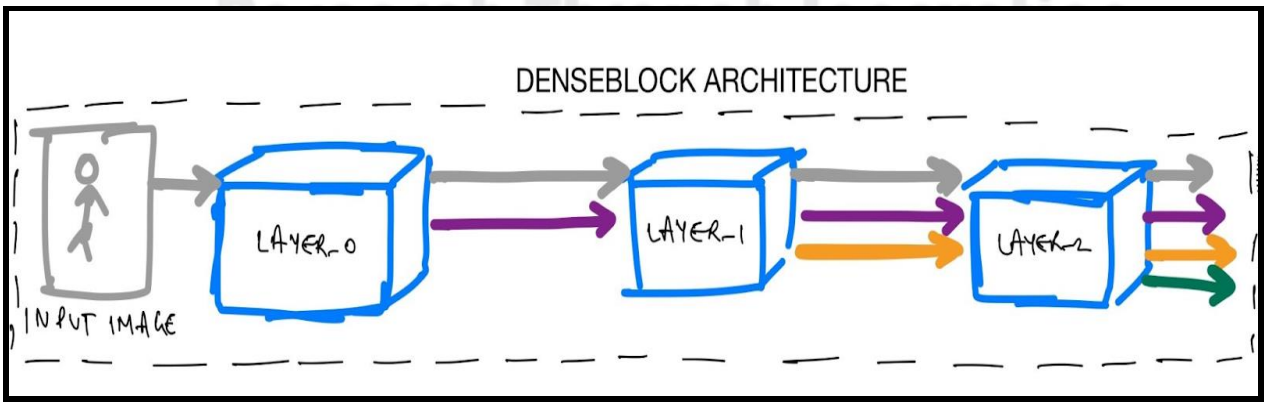
There are 5 Dense Blocks in which all layers are interconnected. First 4 Denseblock are using ReLU as activation function and the last Dense block is using sigmoid as its activation function. Each dense block is connected with a transitional layer.



Architecture of DenseNet

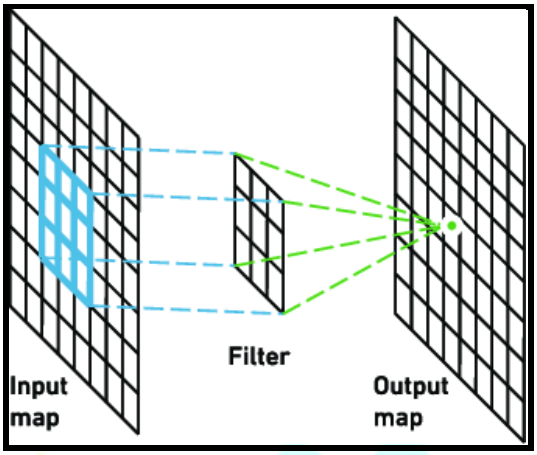
Inside a Dense Block:

Finally, we send some grey input characteristics to LAYER 0. Grey features are modified nonlinearly by LAYER 0 to include purple features. These are then input into LAYER 1, which performs a non-linear adjustment to add orange characteristics in addition to the grey and purple ones. This three-layer dense block eventually combines characteristics in the colours grey, purple, orange, and green.
As a result, each layer in a thick block increases the number of features on the feature maps.
As a result, as you can see, the size of the feature map grows as you move through each dense layer, and new features are concatenated with earlier features. The characteristics of the network include



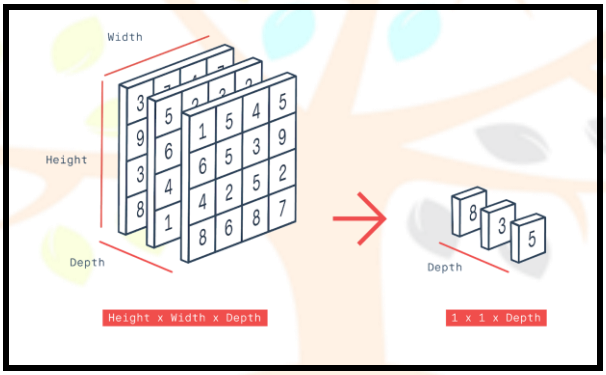
Convolutional Layer:

The convolution layer contains a set of kernels, parameters of which are to be learned throughout the course of training; not each and every parameter is trainable so the size of kernels is typically smaller than the particular image. Each kernel convolves and creates an activation map. The first layer that is utilised to extract information from an input image is convolution.Convolution keeps the relationship between pixels by learning image properties from tiny squares of incoming data. The calculation requires two inputs: an image matrix and a kernel or filter.



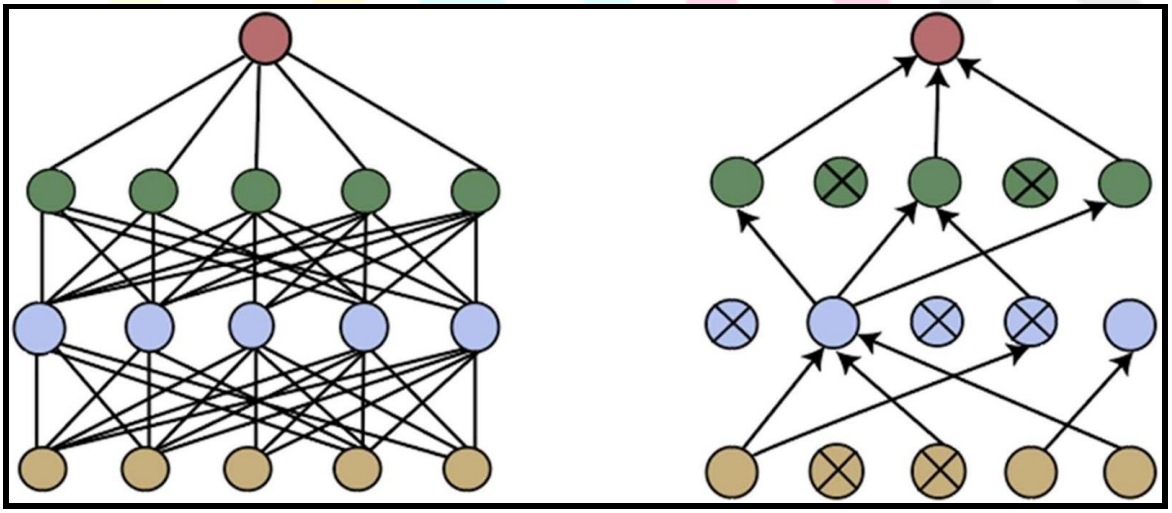
Global Average Pooling Layer:

Global Average Pooling layers are used to decrease the dimensions of the feature maps by taking the average of the region in the feature map. This layer creates a downsampled (pooled) feature map by compiling the features that are present in a specific area of the convolution layer's feature map. This will produce a feature map for each class that corresponds to it in the classification job.



Dropout Layer:

This layer randomly sets input units to 0 at a frequency of rate during each training step to prevent overfitting. In machine learning, the term "dropout" describes the practise of willfully ignoring some nodes in a layer during training.



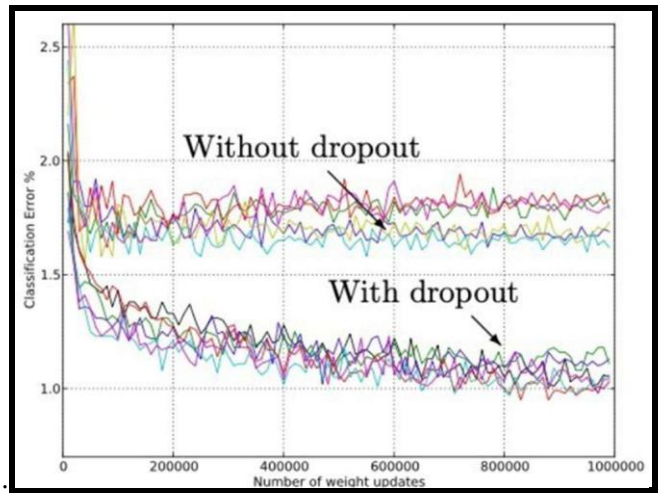
(a)without dropout (b)with dropout

The NN on the left in the image above illustrates a typical NN with all units turned on. The few units on the right have been removed from the model; their weights and biases were not taken into account when training the model.

Why do we need dropouts?

The majority of the parameters are taken into account by a fully connected layer, and as a result, co-dependency between neurons begins to grow during training. This reduces each neuron's individual power and causes the training data to be overly fitted.

Graph comparison between dropout and without Gradropout

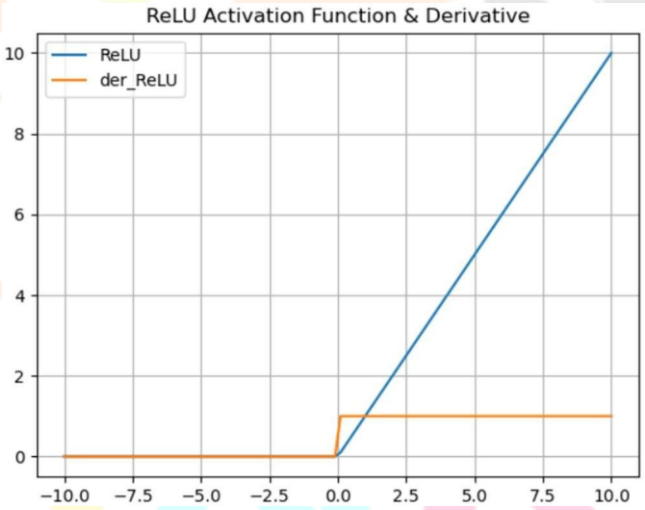


Activation Function:

A NN model can distinguish between outcomes and quickly adjust to different types of input because to the activation function.
If the input is positive, the rectified linear activation function, also known as ReLU, will output the input directly; if the input is negative, it will output zero. As a model that utilises it is simpler to train and frequently outperforms others, it has evolved into the default activation function for many different types of neural networks. The first four dense blocks employ ReLU as their activation function, whereas the final dense block uses Sigmoid.

reLU function is calculated as:

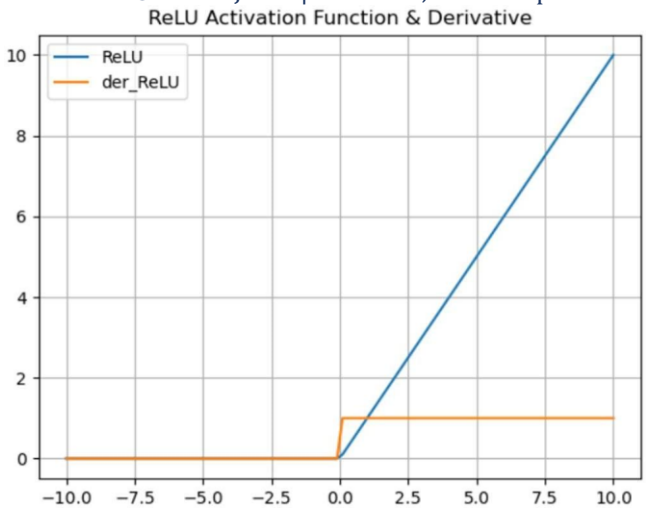
f(x) = x+ = max(0,x)



ReLu Function graph

Calculating the sigmoid activation function yields:

$$f(z) = \frac{1}{1+e^{-z}}$$

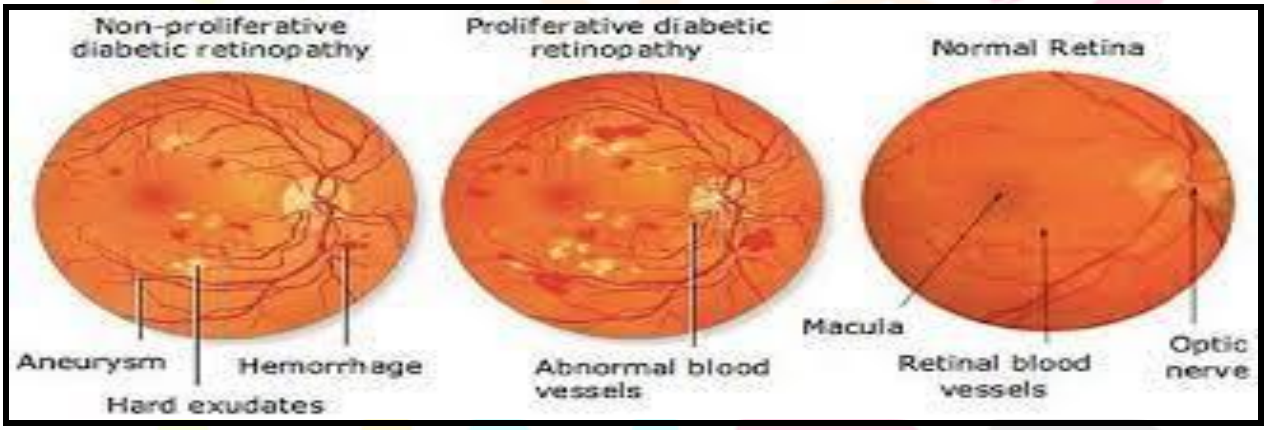


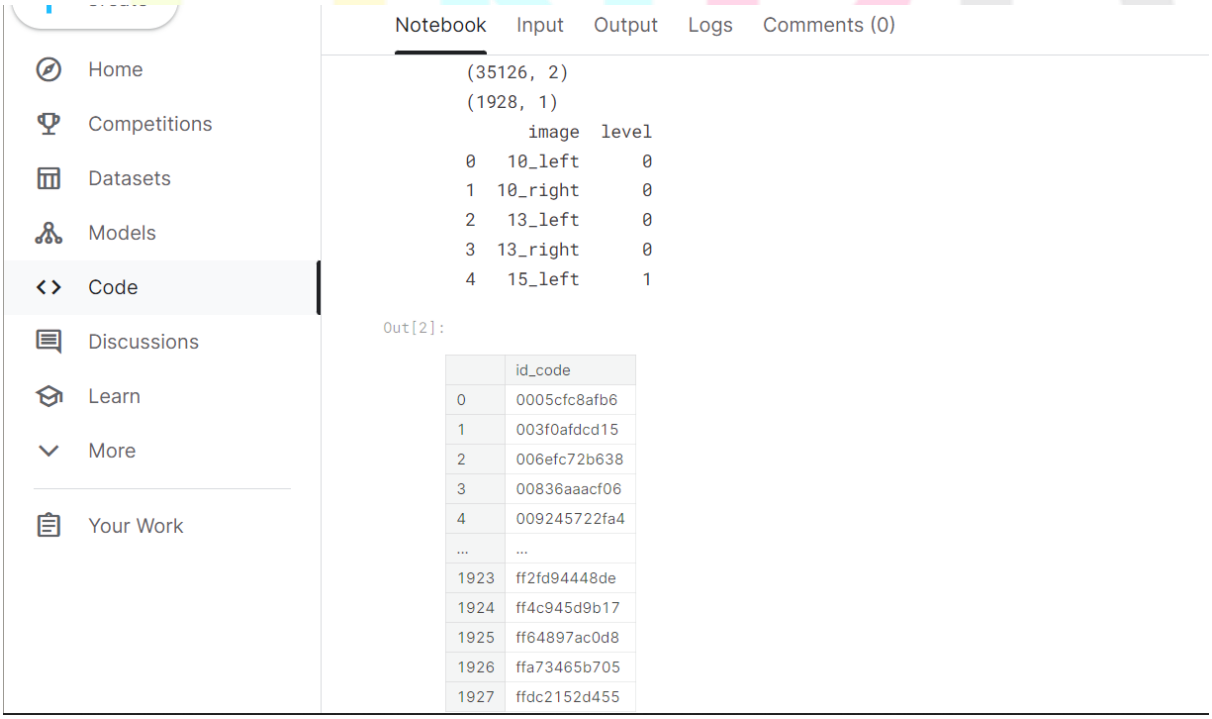
It is used because it exists between the sigmoid function (0 to 1). It is therefore especially utilised for models that call for us to forecast the likelihood as an output. The sigmoid is the best option because anything has a probability that only falls between 0 and 1. Differentiation of a function. This means that any two locations on the sigmoid curve can provide the slope. The derivative of a function is not monotonic, but the function itself is. A neural network's training phase may become stalled due to the logistic sigmoid function.

Activation function helped us tackle many limitations which were possible with using Tanh and softmax functions.

RESULT:

As a result, Densenet can automatically diagnose the complex features involved in classification, such as micro-aneurysms, exudate, and haemorrhages on the retina.



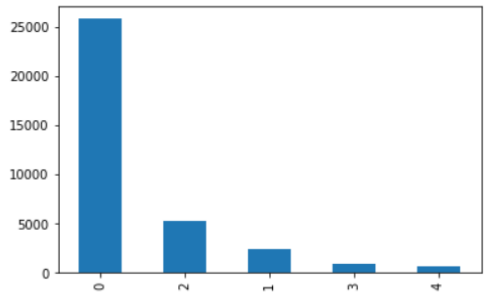


Notebook Input Output Logs Comments (0)

Home
Competitions
Datasets
Models
Code
Discussions
Learn
More
Your Work

In [3]:
```
df_train['level'].value_counts().plot(kind='bar')
df_train.level.value_counts()
```

Out[3]:
```
0    25810
2     5292
1     2443
3      873
4      708
Name: level, dtype: int64
```



Notebook Input Output Logs Comments (0)

Home
Competitions
Datasets
Models
Code
Discussions
Learn
More
Your Work

In [4]:
```
class_weights = df_train['level'].value_counts()
dfs = [df_train[df_train['level'] == i].sample(1200*(2 if i < 3 else 1),replace=True) for i in range(5)]
resampled = pd.concat(dfs, axis = 0).reset_index(drop=True)
resampled
```

Out[4]:

	image	level
0	40740_right	0
1	27179_left	0
2	1422_right	0
3	9277_right	0
4	37301_left	0
...	...	...
9595	35112_right	4
9596	38064_right	4
9597	44247_left	4
9598	25378_right	4
9599	12995_left	4

9600 rows × 2 columns

Notebook Input Output Logs Comments (0)

Home
Competitions
Datasets
Models
Code
Discussions
Learn
More
Your Work

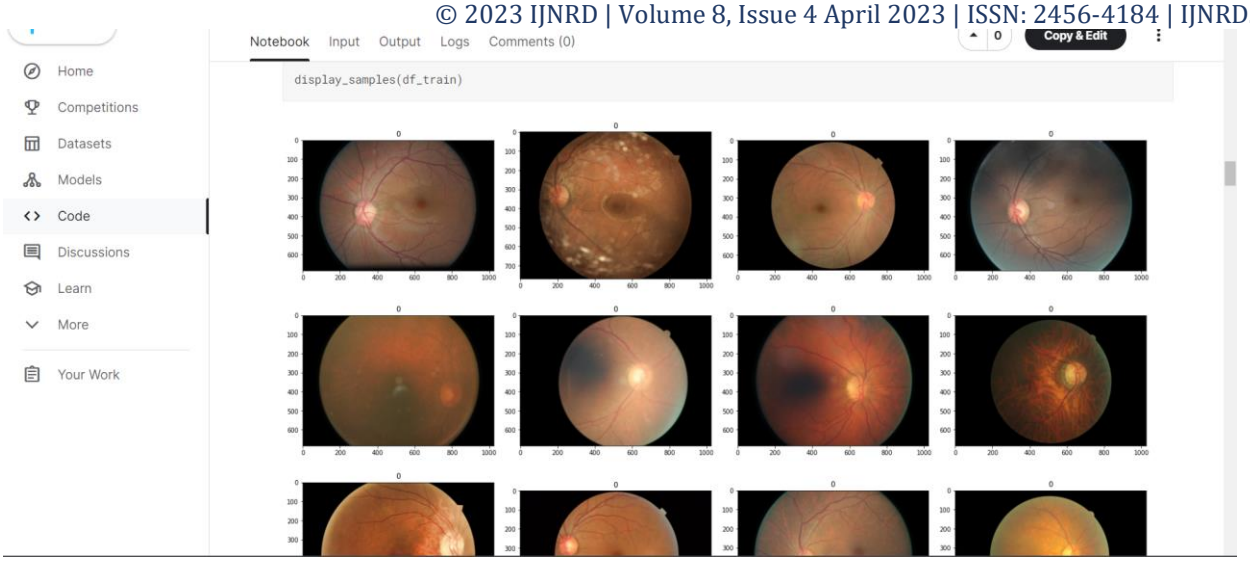
In [5]:
```
resampled.level.value_counts()
```

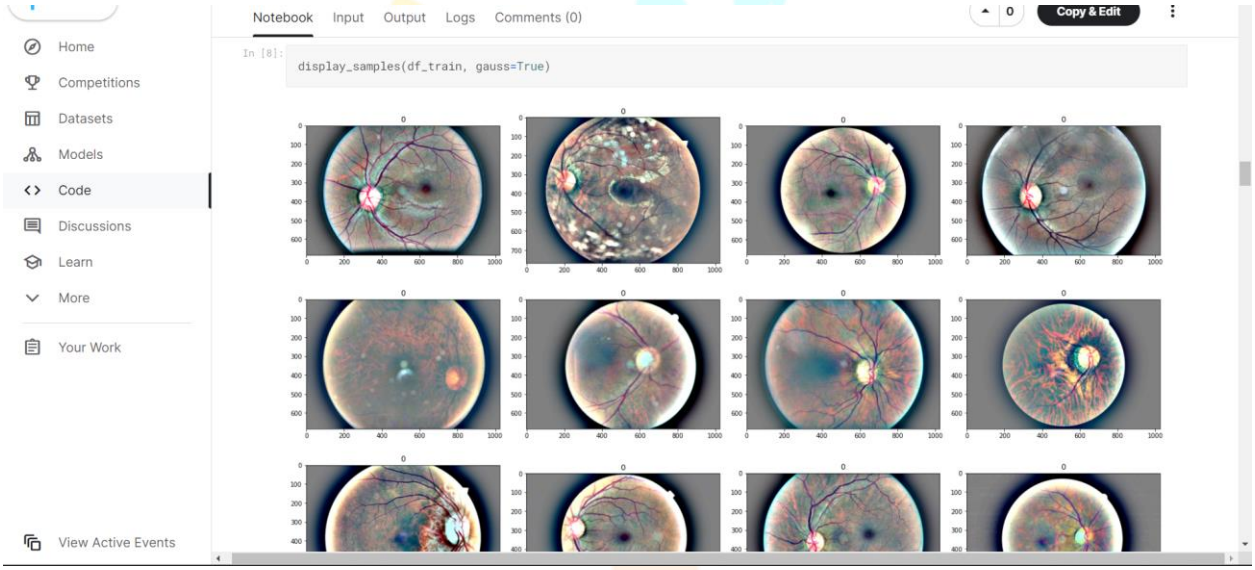
Out[5]:
```
0    2400
1    2400
2    2400
3    1200
4    1200
Name: level, dtype: int64
```

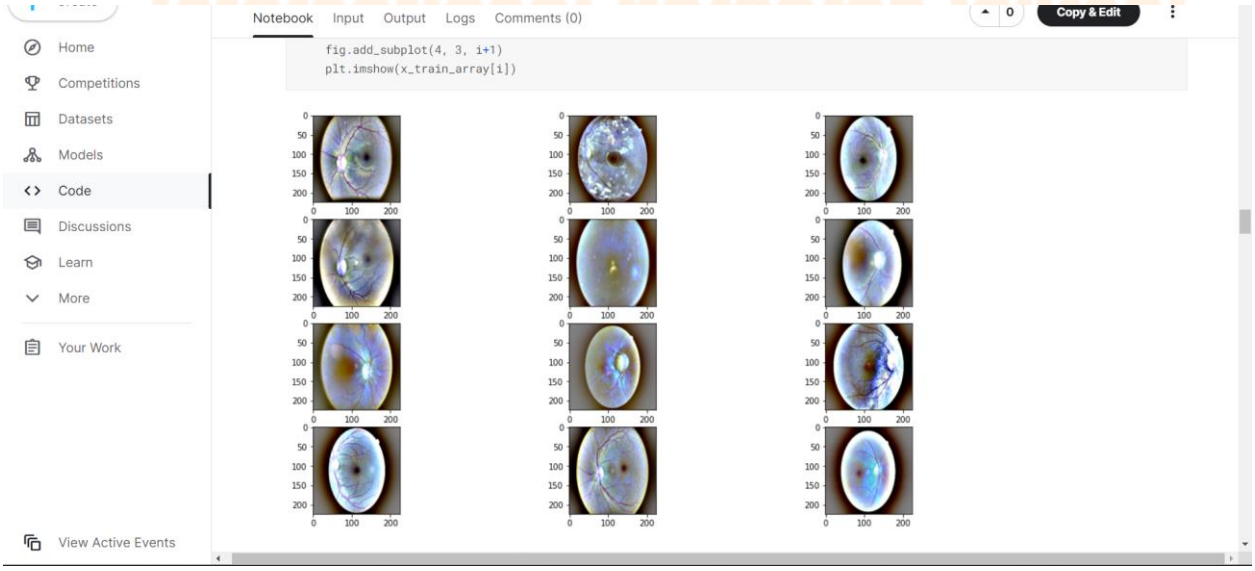
In [6]:
```
df_train=resampled
```

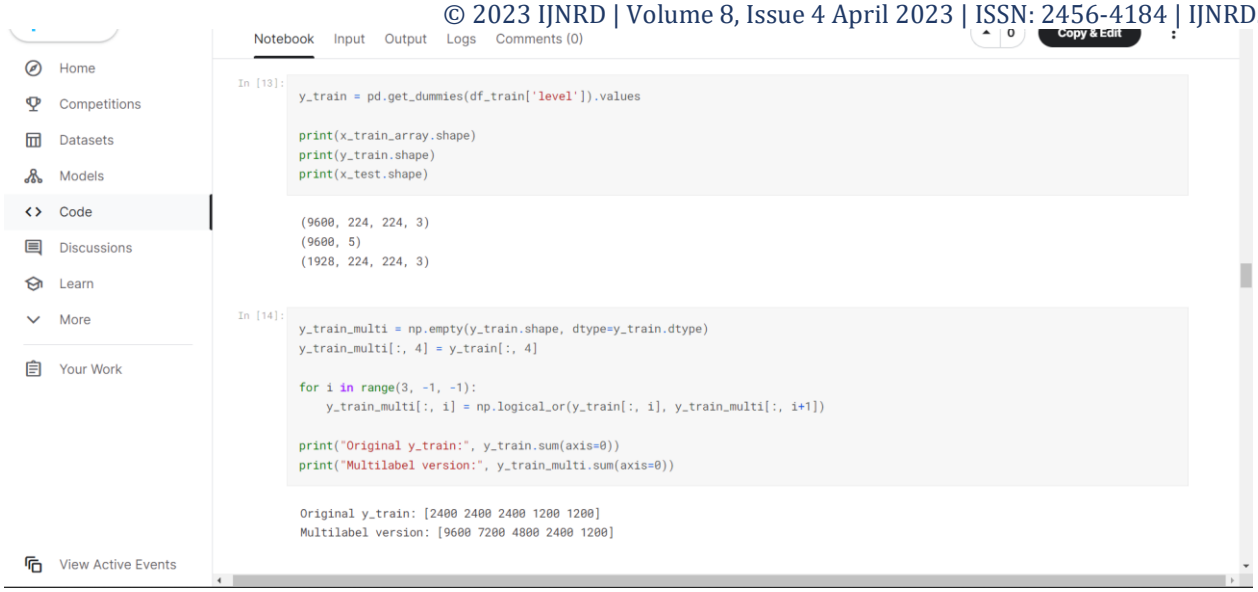
In [7]:
```
def display_samples(df, columns=4, rows=3, gauss=False):
    fig=plt.figure(figsize=(5*columns, 4*rows))

    for i in range(columns*rows):
        image_path = df.loc[i,'image']
```

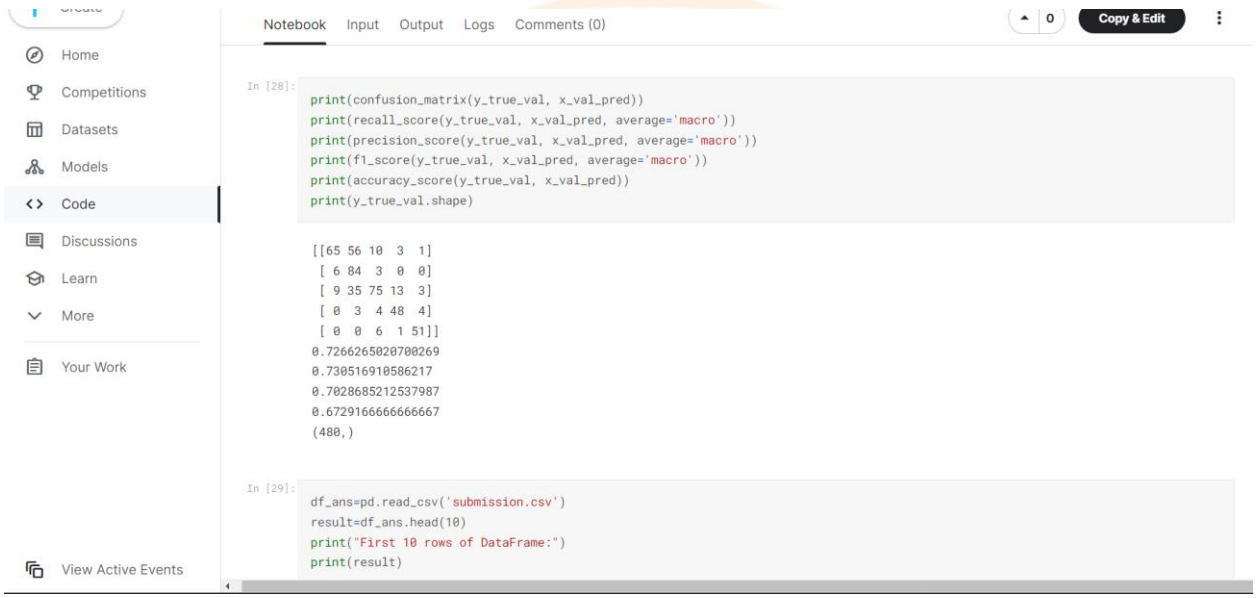

Notebook
Input
Output
Logs
Comments (0)
Copy & Edit
Home
Competitions
Datasets
Models
Code
Discussions
Learn
More
Your Work
display_samples(df_train)


Notebook
Input
Output
Logs
Comments (0)
Copy & Edit
Home
Competitions
Datasets
Models
Code
Discussions
Learn
More
Your Work
View Active Events
In [8]:
display_samples(df_train, gauss=True)


Notebook
Input
Output
Logs
Comments (0)
Copy & Edit
Home
Competitions
Datasets
Models
Code
Discussions
Learn
More
Your Work
View Active Events
fig.add_subplot(4, 3, i+1)
plt.imshow(x_train_array[i])


Notebook Input Output Logs Comments (0)
Copy & Edit
Home
Competitions
Datasets
Models
Code
Discussions
Learn
More
Your Work
View Active Events
In [13]:
y_train = pd.get_dummies(df_train['level']).values
print(x_train_array.shape)
print(y_train.shape)
print(x_test.shape)
(9600, 224, 224, 3)
(9600, 5)
(1928, 224, 224, 3)
In [14]:
y_train_multi = np.empty(y_train.shape, dtype=y_train.dtype)
y_train_multi[:, 4] = y_train[:, 4]
for i in range(3, -1, -1):
y_train_multi[:, i] = np.logical_or(y_train[:, i], y_train_multi[:, i+1])
print("Original y_train:", y_train.sum(axis=0))
print("Multilabel version:", y_train_multi.sum(axis=0))
Original y_train: [2400 2400 2400 1200 1200]
Multilabel version: [9600 7200 4800 2400 1200]


Notebook Input Output Logs Comments (0)
Copy & Edit
Home
Competitions
Datasets
Models
Code
Discussions
Learn
More
Your Work
View Active Events
In [27]:
from sklearn.metrics import confusion_matrix, recall_score, precision_score, f1_score, accuracy_score
print(confusion_matrix(y_true_train, x_train_pred))
print(recall_score(y_true_train, x_train_pred, average='macro'))
print(precision_score(y_true_train, x_train_pred, average='macro'))
print(f1_score(y_true_train, x_train_pred, average='macro'))
print(accuracy_score(y_true_train, x_train_pred))
print(y_true_train.shape)
[[1971 290 4 0 0]
[16 2281 10 0 0]
[10 115 2123 13 4]
[0 1 4 1121 15]
[0 0 1 3 1138]]
0.9550408720652891
0.9593036954397295
0.9552147199284555
0.9467105263157894
(9120,)
In [28]:
print(confusion_matrix(y_true_val, x_val_pred))
print(recall_score(y_true_val, x_val_pred, average='macro'))


Notebook Input Output Logs Comments (0)
Copy & Edit
Home
Competitions
Datasets
Models
Code
Discussions
Learn
More
Your Work
View Active Events
In [28]:
print(confusion_matrix(y_true_val, x_val_pred))
print(recall_score(y_true_val, x_val_pred, average='macro'))
print(precision_score(y_true_val, x_val_pred, average='macro'))
print(f1_score(y_true_val, x_val_pred, average='macro'))
print(accuracy_score(y_true_val, x_val_pred))
print(y_true_val.shape)
[[65 56 10 3 1]
[6 84 3 0 0]
[9 35 75 13 3]
[0 3 4 48 4]
[0 0 6 1 51]]
0.7266265020700269
0.730516910586217
0.7028685212537987
0.6729166666666667
(480,)
In [29]:
df_ans=pd.read_csv('submission.csv')
result=df_ans.head(10)
print("First 10 rows of DataFrame:")
print(result)