# Astronomical Images Classification Using Deep Learning CNNs

**Article** · February 2021

DOI: 10.6084/m9.figshare.13720948.v1



**1 author:**

Yosry Negm
Menoufia University
**4** PUBLICATIONS **0** CITATIONS

SEE PROFILE

# Astronomical Images Classification Using Deep Learning CNNs

Yosry Negm, Faculty of Electronic Engineering, Menoufia University, Egypt

Project associated with the paper is located at https://github.com/yosrinegm/Astronomical-Images-Classification

# 1. Abstract

## 1.1. Overview

Recently, Astronomy has been witnessed a great jump and advancements in detectors, instruments, telescopes and even the probes that are sent into outer space and far planets for collecting data in sky surveys to map our universe. These collected data are organized into very large datasets forming what is called data-oriented astronomy. Unfortunately, it is impossible to work on these massive datasets manually to get effective results so, astronomers are seeking approaches to automate the human error borne processes of manual scanning in order to discover astronomical knowledge and information from these large raw datasets through state of the art data mining techniques, statistical methods and data science tools (This a new discipline called *Astroinformatics*[1]). I Think, as many others that there is no better player here than applying machine learning algorithms and techniques on massive astronomical datasets to classify stars, quasars, and galaxies either by using photometric images, redshifts or radio astronomy collected signals datasets. Historically, early efforts started in 2010 as initiatives in National Research Council (United states)[2]. This step was the basis on which later research contributions[3] focused on and enriched the field using large worldwide distributed collection of digital astronomical databases such as Large Synoptic Survey Telescope (LSST), Square Kilometer Array observatory, and Sloan Digital Sky Survey (SDSS). But the field is still young and need more research to find new precise techniques especially in machine learning and specifically utilizing deep learning and availability of massive astronomical datasets opened now for public. Theses classification problems should be solved in order to correctly map our universe to gain more understanding about it and endorse the existing and new theories in cosmology, of course the perfect candidate technique here is deep learning, since it has proved its success on massive image datasets such as the case of our domain. From this point of view, I've selected concrete astronomical classification problem to investigate applying a Deep Learning algorithm in order to validate this mentioned possibility using the well-known public dataset of the Sloan Digital Sky Survey (**SDSS**)[4] Supernova surveys specifically the SDSS legacy survey named SDSS-II SN Survey.

[1] https://en.wikipedia.org/wiki/Astroinformatics
[2] Astronomy and Astrophysics Decadal Survey ( https://www.nap.edu/read/12951/chapter/1 )
[3] Examples of related Academic Research : http://iopscience.iop.org/article/10.1086/507440 , https://arxiv.org/abs/astro-ph/0106481 , https://arxiv.org/abs/1711.05744 and https://arxiv.org/abs/1706.02467 .
[4] https://classic.sdss.org/supernova/aboutsupernova.html


## 1.2. Problem statement

Sky surveys provides massive astronomical datasets in many different forms such as optical spectra, infrared spectra, photometric redshifts, light curves and imaging data which represent variety of astronomical information and objects. Considering imaging data, astronomers start the classification process manually by scanning the in hand images to detect what is real and what is not in then group them into groups of possible stars, quasars, and galaxies. Imaging data often contains objects that appear for a short amount of time, called Transients such as gamma rays, asteroids, and supernovae. Supernova[5] (SN) occurs at the end of star's life time i.e. when a star is destroyed by an explosion, it will extremely shine throughout the galaxy. The resulting supernovae (SNe) are classified based on their spectral features at maximum light into five main categories (Type I, Type II, Type Ib, Type Ic, Type Ia) differs in the presence of Hydrogen, Helium and other elements in their spectra. These supernovae explosions are either thermonuclear explosions of white dwarfs (for stars around 1.4 solar mass and dipoles) or core collapse of the star (massive stars around 9 solar mass) which releases gravitational energy and remains as neutron stars (pulsars) or black holes. The problem here that the classification of artefacts and objects for these physical phenomena is done by hand and it is a very time consuming job and also it is subject to human bias which differs from person to person, as well as the manual scanning is infeasible for a huge amount of images. As a solution to for this problem, there are grand efforts to explore the possibility of applying machine learning algorithms to automate the process of knowledge discovery and the extraction of astronomical information within these large raw datasets instead of the infeasible manual and human handling. Herein our project we have investigated the applying of a Deep Learning algorithm in order to validate this mentioned possibility but for the purpose of the comparison to our benchmark model I have chosen the astronomers' first step which is looking for interesting objects in a binary classification problem of determining what is real and what is not in the underlying imaging data.

[5] https://spaceplace.nasa.gov/supernova/en/

Human hand scanning classification for the objects in the dataset

| | Original_Classes | Visual_Classes | Binary_Classes | Description |
|---|---|---|---|---|
| **0** | Artefact | Artefact | Not-Real | Residuals caused by problems in the image |
| **1** | Dipole | Dipole/Saturated | Not-Real | Residuals caused by errors in image registration |
| **2** | Saturated Star | Dipole/Saturated | Not-Real | Residuals of stars that saturate the CCD |
| **3** | Moving | Real | Real | Anything showing signs of motion |
| **4** | Variable | Real | Real | Objects showing a record of long-term variability |
| **5** | Transient | Real | Real | Objects with no observation history |
| **6** | SN Other | Real | Real | Objects that are thought to be SNe |
| **7** | SN Bronze | Real | Real | residuals at the centre of their host galaxies |
| **8** | SN Silver | Real | Real | SNe much more luminous |
| **9** | SN Gold | Real | Real | Possible SNe identified |

## 1.3. Metrics

For evaluating the results of my work and comparing it to the selected benchmark model I've used the same four metrics of performance measure that used there. The four measures that are used in this comparison are commonly applied to classification problems and they are **A**ccuracy (A), **P**recision (P), **R**ecall (R), and **$F_1$-Score**. They are defined in terms of true/false positives/negatives ($t_p$, $t_n$, $f_p$, $f_n$). where positive classes are those corresponding to real objects. we also introduced some other metrics such as **Confusion Matrix** and area under curve (**AUC**) as well as receiver operating characteristic (**ROC**) curve which is a plot of true positive rate (**TPR**) versus false positive rate (**FPR**) at various threshold values to study how the discrimination threshold of our binary classifiers is varied and demonstrate the trade-off between false positives and false negatives**.**

**Accuracy**

Accuracy is defined as the fraction of predictions our model got right, it could be a reasonable initial measure since the classes in our dataset are all of similar sizes and is computed as follows: -

$$A = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions}$$

$$A = \frac{t_p + t_n}{t_p + t_n + f_p + f_n}$$

*Where $T_P$ = True Positives, $T_n$ = True Negatives, $F_P$ = False Positives, and $F_n$ = False Negatives.*

**Precision**

Precision reflects the fraction of reported real objects that are so (proportion of positive identifications was actually correct). It is calculated from the following formula: -

$$P = \frac{t_p}{t_p + f_p}$$

**Recall (**True Positive Rate or **TPR)**

Recall as the fraction of true objects that are found by the classifier (proportion of actual positives was identified correctly). It is calculated as follows: -

$$P = \frac{t_p}{t_p + f_n}$$

**$F_1$-Score**

$F_1$-Score is a measure of a test's accuracy and is expressed in terms of Precision and Recall (Harmonic Mean between precision and recall), it could be considered as a measure that punishes false negatives and false positives equally but weighted by their inverse fractional contribution to the full set to account for large class number hierarchies. It computed as follows: -

$$F_1 = 2 * \frac{1}{\frac{1}{Precision} + \frac{1}{Recall}} \qquad or \qquad F_1 = \frac{2PR}{P + R}$$

**False Positive Rate (**Fall-out**)**

$$FPR = \frac{f_p}{f_p + t_n}$$

**Confusion Matrix**

Confusion Matrix contains the counts of the occurrences of all the possible model prediction outcomes, for every classification there are four possible outcomes: If the model correctly predicts "Real" object it is called True Positive (Tp) and if incorrectly classed as "Not Real" objects then it is a false negative (Fn). On the other hand, if the model correctly predicts the object to be "Not-Real" it is a true negative (Tn). But if classed as a "Real" when it is not, then it is a false positive (Fp). Briefly, It contains the total numbers true/false positives/negatives. And could be constructed as shown below:

| | | **Predicted Classes** | |
|---|---|---|---|
| | | Real | Not-Real |
| **True classes** | Real | $t_p$ | $f_n$ |
| | Not-Real | $f_p$ | $t_n$ |

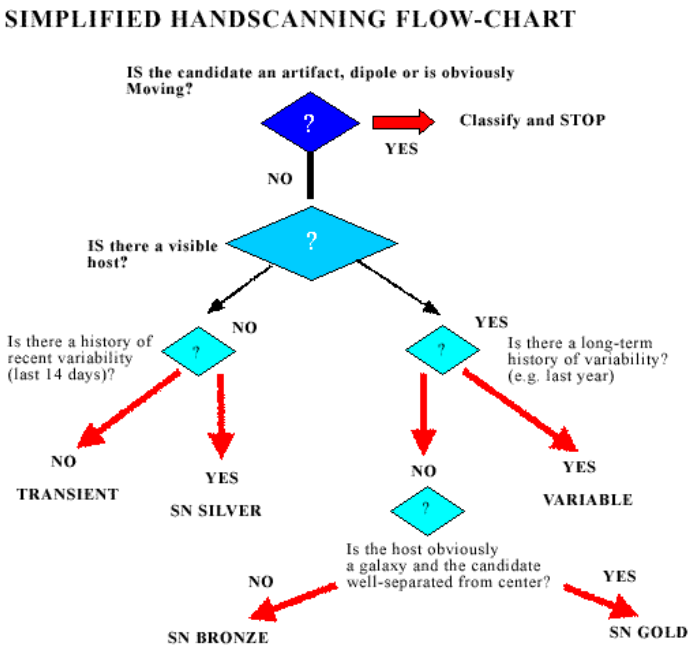
**Area Under Curve** (Area Under the ROC curve or AUC)

It is a statistic that often used for model comparison in machine learning community, and it is equal to the probability that a classifier will correctly classify a randomly chosen instance. And it is a common way of comparing models to each other in a single measure.

# 2. Analysis

## 2.1. Data exploration

The dataset used in this project is one of the data releases found on SDSS Supernova surveys specifically the SDSS legacy survey named SDSS-II SN Survey, taken at 300 deg equatorial stripe 82. This dataset release was obtained between 2005 and 2008 but it is a well calibrated uniform map of the universe, which still suitable to be used for scientific studies for decades to come. It contains around 21,785 images of size 69x69 pixels and it is manually[6] classified into ten classes: -



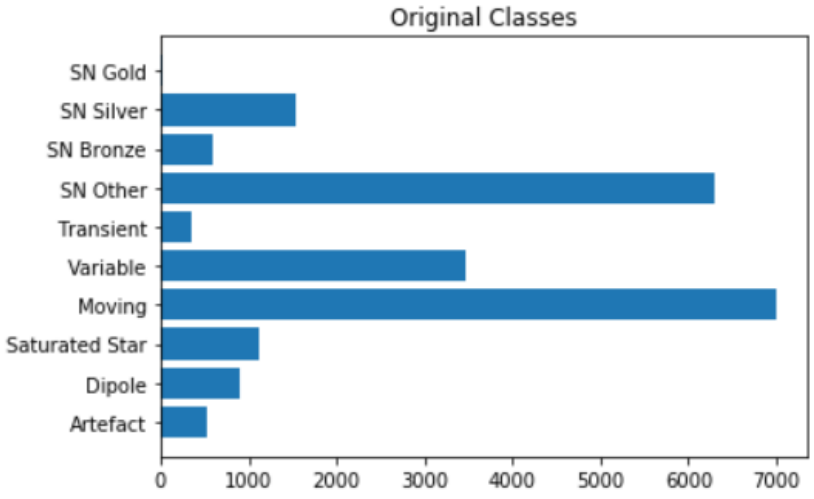

[6] Hand scanning guide : http://spiff.rit.edu/richmond/sdss/sn_survey/scan_manual/sn_scan.html

Astronomers are using the above flow chart as a manual decision tree in their hand scanning of the collected imaging data which gives the following original classes: -

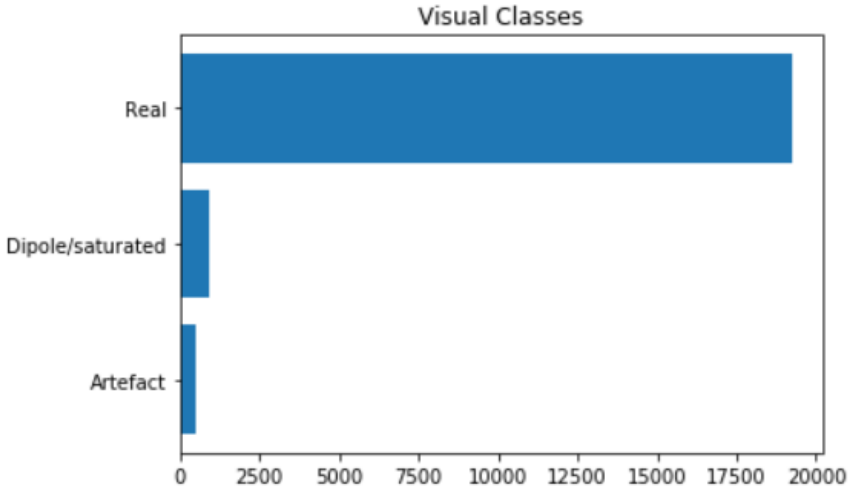
- **Artefact**: Residuals caused by problems in the image (e.g. diffraction spikes), this is considered **Not-Real**.
- **Dipole**: Residuals with roughly equal amounts of positive and negative flux, caused by errors in image registration, this is considered **Not-Real**
- Saturated star: Residuals of stars that saturate the CCD, this is considered **Real**
- **Moving**: Anything showing signs of motion between cutouts in different passbands, this is considered **Real**
- **Variable**: Objects showing a record of long-term variability, this is considered **Real**
- **Transient**: Objects with no observation history, no apparent host galaxy in the search image and no motion, this is considered **Real**
- **SN Other**: Objects that are thought to have a good chance of being SNe, but that do not fit nicely into any of the above classes, this is considered **Real**
- **SN Bronze**: Point-like residuals at the center of their host galaxies, most of the objects in this class later turn out to be either quasars (QSOs), active galactic nuclei (AGN) or, foreground variable stars, and not SNe, this is considered **Real**
- **SN Silver**: Point-like residuals having no apparent host galaxy, SNe much more luminous than their host galaxies usually fall into this class, this is considered **Real**
- **SN Gold:** Possible SNe identified as point-like residuals that are not at the exact center of their host galaxies, this is considered **Real**

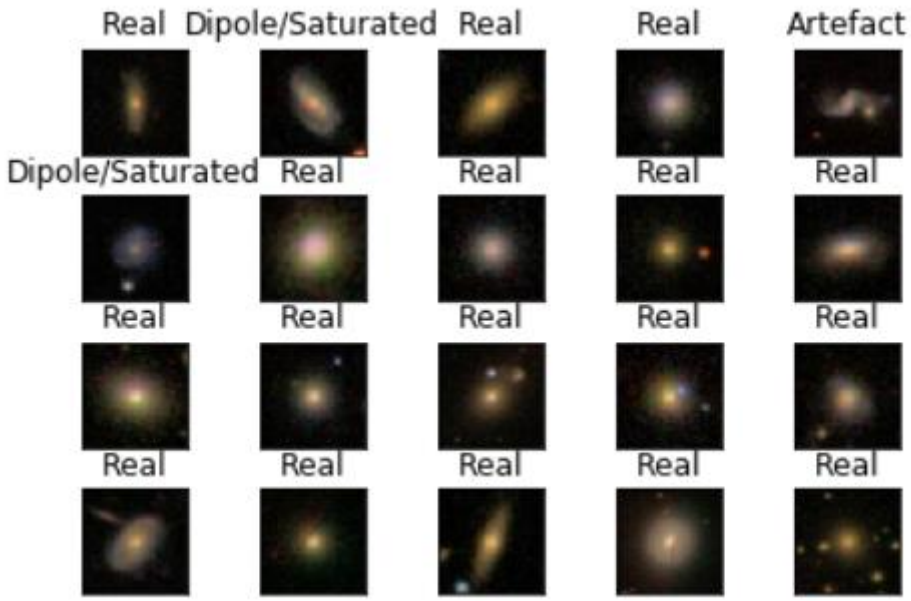
## 2.2. Exploratory visualization

The ten classes (dipoles, artefacts, saturated stars, transients, variables, moving objects, SN Gold, SN Silver, SN Bronze, SN Other) are called original classes, the following graph explores their totals in the or dataset.

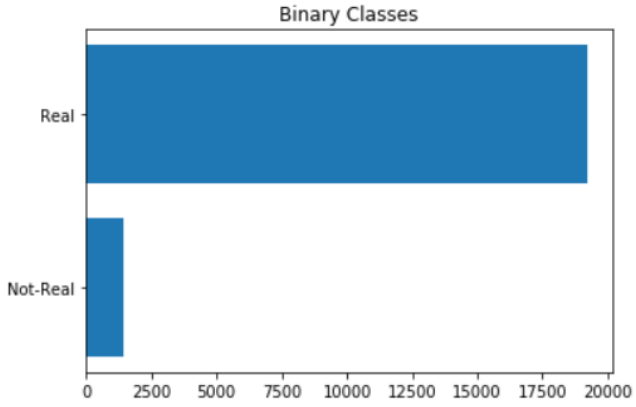


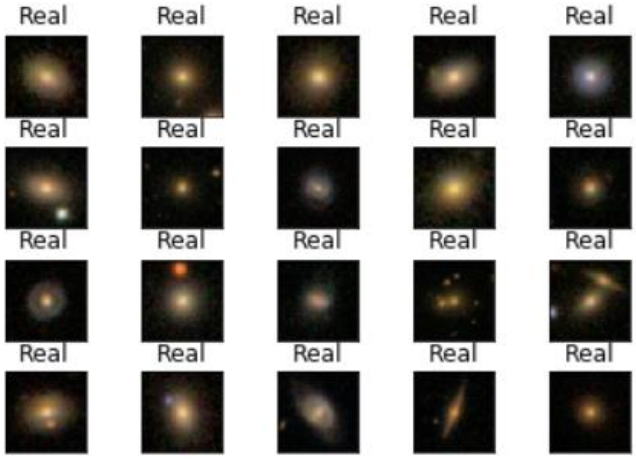

For simplification purpose, the dataset is reduced to three visual classes (**Real**, **Artefacts**, **Saturated/ Dipole**) as shown below.



To compare with our benchmark model, we considered a binary classification of determining what is **real** and what **is not** in the underlying imaging data, the following graph explains the manual labeling statistics for **Real/Not-Real** classification of the data points. There are 88.24 % of the data are classified as Real by human scanners.




## 2.3. Algorithms and techniques

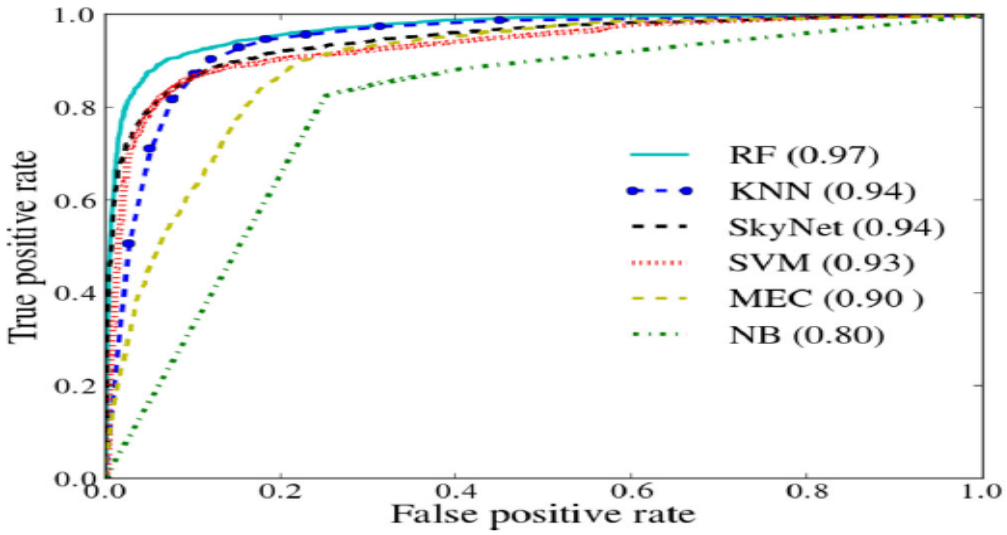
In order to reduce human bias depending on the mood and tiredness of who perform manual scanning, SDSS datasets is injected randomly with some amount of fake objects images among the original images. The remaining part of the problem is the infeasibility of classifying hundreds and thousands of images per night by human, so they should be replaced by machines and this what I am trying to validate in my project. I will construct a convolutional neural network to automate the manual process of classifying transient imaging data from SDSS Supernova survey into real objects and artefacts since this algorithm provides impressive image recognition results. Difference images are created by subtracting a reference image from the most recent image of a given part of the sky, this will leave a pure noise image unless transient[7] is exist. To simplify our problem, I will convert it to binary classification task i.e. the mission will be to detect real transients from non-real ones[8].

[7] **Transients** are objects that appear for a short amount of time such as supernova, asteroids, variable star, or others.
[8] http://spiff.rit.edu/richmond/sdss/sn_survey/scan_manual/sn_scan.html

## 2.4. Benchmark

Most of the similar works to this project had been focused on SN Factory[9] prior work (Roman, Aragon & Ding in 2006 and Bailey 2007) utilizing classical supervised learning algorithms to accomplish the task of automatic classification. As a benchmark model for my work, I will suggest the publication titled " Machine learning classification of SDSS transient survey images "[10] . This research works on the same dataset, but used different learning algorithms such as Random forest, k-nearest neighbors, Naïve Bayes, and Support vector machine (SVM). Then it compares their performance using the same measure metrics I have used, in other words I had reworked this research but using deep learning convolutional networks and then compared my work to the prior work. But on contrast they depend on Principal Component Analysis Algorithm (PCA) to extract features like attributes as shape, position, Full-Width Half Maximum (FWHM), and distance to the nearest object in the difference image as well as SDSS Camera filters (g, r, i, z, u)[11], In our solution there is no need to process feature extraction by using PCA but our CNN will learn them. The benchmark model had achieved the following results which I am comparing my results to them.

| Machine learning technique | AUC | Accuracy | Recall | Precision | F1-score | Confusion matrix | Object | Not object |
|---|---|---|---|---|---|---|---|---|
| Random forest (RF) Section 5.6 | **0.97** | **0.91** | 0.91 | **0.93** | **0.92** | Object<br>Not object | 3541<br>259 | 342<br>2732 |
| *k*-nearest neighbours (KNN) Section 5.3 | 0.94 | 0.89 | 0.90 | 0.91 | 0.90 | Object<br>Not object | 3506<br>363 | 377<br>2628 |
| SkyNet Section 5.5 | 0.94 | 0.88 | 0.89 | 0.90 | 0.89 | Object<br>Not object | 3461<br>399 | 422<br>2592 |
| Support vector machine (SVM) Section 5.4 | 0.93 | 0.86 | 0.90 | 0.85 | 0.87 | Object<br>Not object | 3514<br>605 | 369<br>2386 |
| Minimum error classification (MEC) Section 5.1 | 0.90 | 0.84 | **0.92** | 0.83 | 0.87 | Object<br>Not object | 3559<br>754 | 324<br>2237 |
| Naïve Bayes (NB) Section 5.2 | 0.80 | 0.77 | 0.86 | 0.77 | 0.81 | Object<br>Not object | 3333<br>998 | 550<br>1993 |




[9] https://snfactory.lbl.gov/
[10] https://academic.oup.com/mnras/article/454/2/2026/1051683
[11] http://skyserver.sdss.org/dr2/en/proj/advanced/color/sdssfilters.asp , https://www.sdss.org/instruments/camera/

# 3. Methodology

## 3.1. Data processing

For unbiased comparison, I have split the dataset into 25% as test set, 75% for training (30% of them are used for cross validation).

| Item | Total count | Percentage % |
|---|---|---|
| Input data set | 21,785 images | - |
| Original classes | 10 classes | - |
| Visual Classes | 3 classes | - |
| Binary classes | 2 Output labels | - |
| Test Dataset | 5,447 | 25 % |
| Training Dataset | 11,436 | 52.49 % |
| Validation Dataset | 4,902 | 22.5 |

Before using the input dataset in our model I have done preprocessing on the data to be suitable for Keras, such as converting the PIL images into 3D tensors then converting these tensors in turn to 4D tensors with the appropriate shape.

| Item | Dimensions |
|---|---|
| Training Tensors | 11436 x 69 x 69 x 3 |
| Validation Tensors | 4902 x 69 x 69 x 3 |
| Test Tensors | 5447 x 69 x 69 x 3 |
| Training Labels | 11436 x 2 |
| Validation Labels | 4902 x 2 |
| Test Labels | 5447 x 2 |

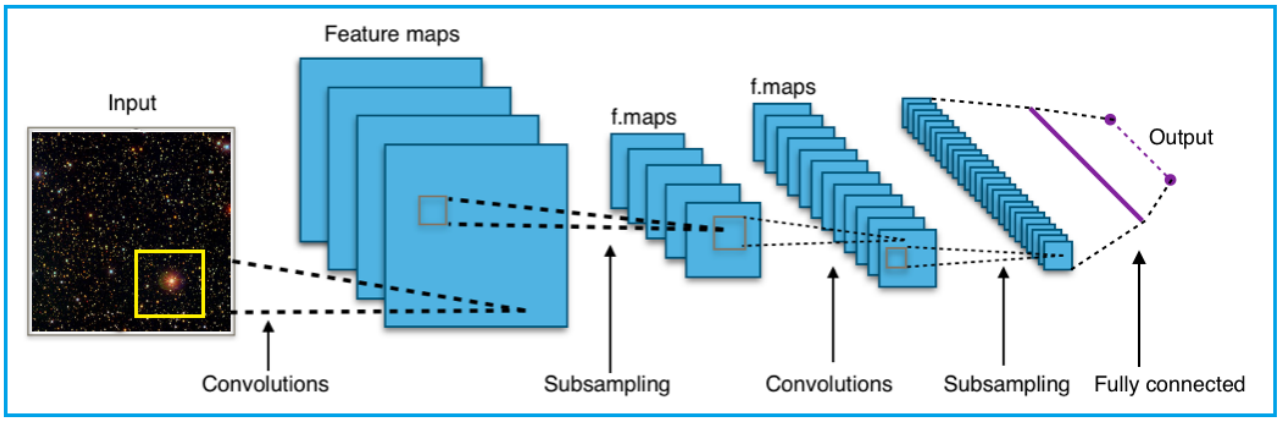
In addition, I have performed **Rescaling** the images' pixel values to be of float32 values that ranges (from 0 to 1) instead of (from 0 to 255).

After that I have converted the ten Original classes to binary classes (Real and Not-Real objects) to be our output class labels and then converting the categorical data of them into a vector of numbers using One Hot Encoding (**OHE**). For the problem statement, the one hot encoding is a row vector, and for each image, it is of a dimension of 1 x 2. The important thing to note here is that the vector consists of all zeros except for the class that it represents, and for that, it is 1.

| | 0 | 1 |
|---|---|---|
| **0** | 0.0 | 1.0 |
| **1** | 0.0 | 1.0 |
| **2** | 0.0 | 1.0 |

## 3.2. Implementation

To solve the proposed problem, I have built the learning model using the deep learning algorithm which was inspired by the biological visual cortex in human brain named deep feed-forward convolutional neural network (**CNN** or ConvNet). I have implemented it in **Python** programming language with **Keras** inside Jupiter Notebook (IPython) depending on using Numpy, Pandas, and Scikit-Learn packages. The reason behind choosing this algorithm is that it reduces the number of parameters and extracts highly robust non-linear features. As well as, it learns hierarchical representations of the data using local receptive fields, sparse connectivity, sharing weights, pooling, and deep architecture. We constructed this network by adding many layers, some for feature representations known as feature maps. After building, compiling, training and evaluating our model we have tuned its hyper-parameters, then re-train and re-evaluate it again then compared our final results to the bench mark model.




As shown above we the image is fed as an input to the network, which goes through multiple convolutions, subsampling, a fully connected layer and finally outputs the image class. The convolution layer computes the output of neurons that are connected to local regions or receptive fields in the input, each computing a dot product between their weights and a small receptive field to which they are connected to in the input volume. Each computation leads to extraction of a feature map from the input image. The objective of subsampling is to get an input representation by reducing its dimensions, which helps in reducing overfitting. One of the techniques of subsampling is max pooling. With this technique, you select the highest pixel value from a region depending on its size. In other words, max pooling takes the largest value from the window of the image currently covered by the kernel. Finally, the objective of the fully connected layer is to flatten the high-level features that are learned by convolutional layers and combining all the features. It passes the flattened output to the output layer where we used a **softmax** activation function to predict the input image class label.

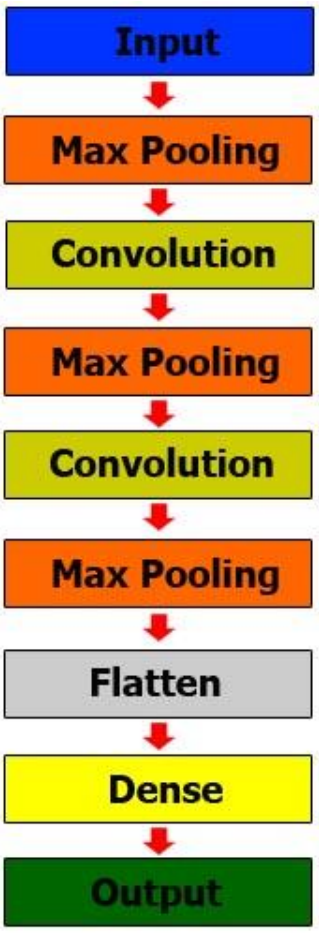
Our solution started with initial model of CNN Network constructed of three convolutional layers as follows:

- The first layer contains 32 filters of size 3 x 3,
- The second layer contains 64 filters of size 3 x 3, and
- The third layer contains 128 filters of size 3 x 3.

In addition, there are three max-pooling layers each of size 2 x 2.

| Layer (type) | Output Shape | Param # |
|---|---|---|
| conv2d_18 (Conv2D) | (None, 69, 69, 32) | 896 |
| max_pooling2d_9 (MaxPooling2 | (None, 35, 35, 32) | 0 |
| conv2d_19 (Conv2D) | (None, 35, 35, 64) | 18496 |
| max_pooling2d_10 (MaxPooling | (None, 18, 18, 64) | 0 |
| conv2d_20 (Conv2D) | (None, 18, 18, 128) | 73856 |
| max_pooling2d_11 (MaxPooling | (None, 9, 9, 128) | 0 |
| flatten_3 (Flatten) | (None, 10368) | 0 |
| dense_5 (Dense) | (None, 128) | 1327232 |
| dense_6 (Dense) | (None, 2) | 258 |

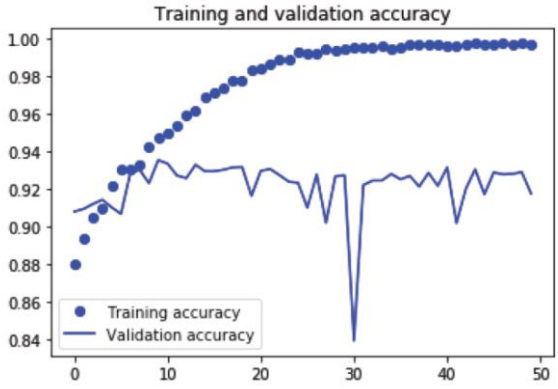
Total params: 1,420,738
Trainable params: 1,420,738
Non-trainable params: 0

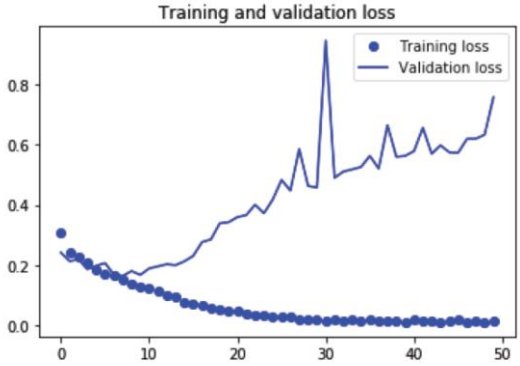


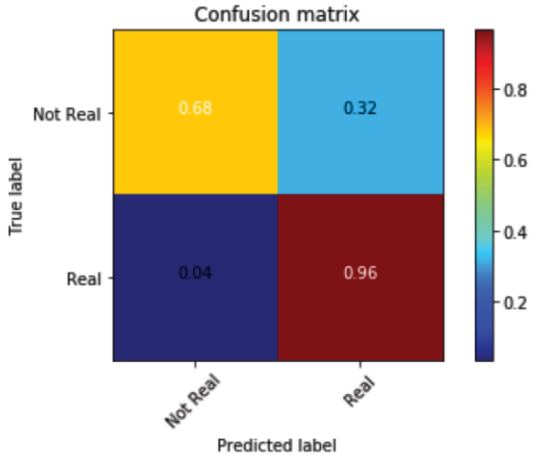

After all convolutional and dense layers, I have use activation function of type "RelU" except for the output layer I have used "Sigmoid". The weight update method here is done by compiling the model using **RMSprop** optimizer, **Batch size equal to 64** and **number of epochs was 50**, I have used the RMSprop optimizer, one of the most popular optimization algorithms which is similar to the gradient descent algorithm with momentum, and it restricts the oscillations in the vertical direction Therefore, we can increase our learning rate and our algorithm could take larger steps in the horizontal direction converging faster. Additionally, I specified the loss type to be **binary cross entropy** and the metrics as accuracy.

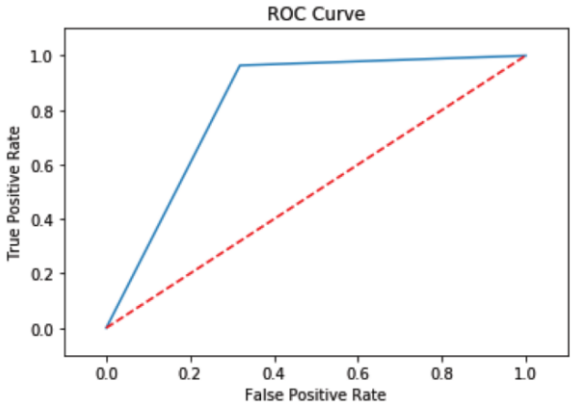
This mentioned setup yields the following results:

| Accuracy | Recall | Precession | $F_1$-score |
|---|---|---|---|
| 93.13 % | 96.43 % | 95.83 % | 96.13 % |



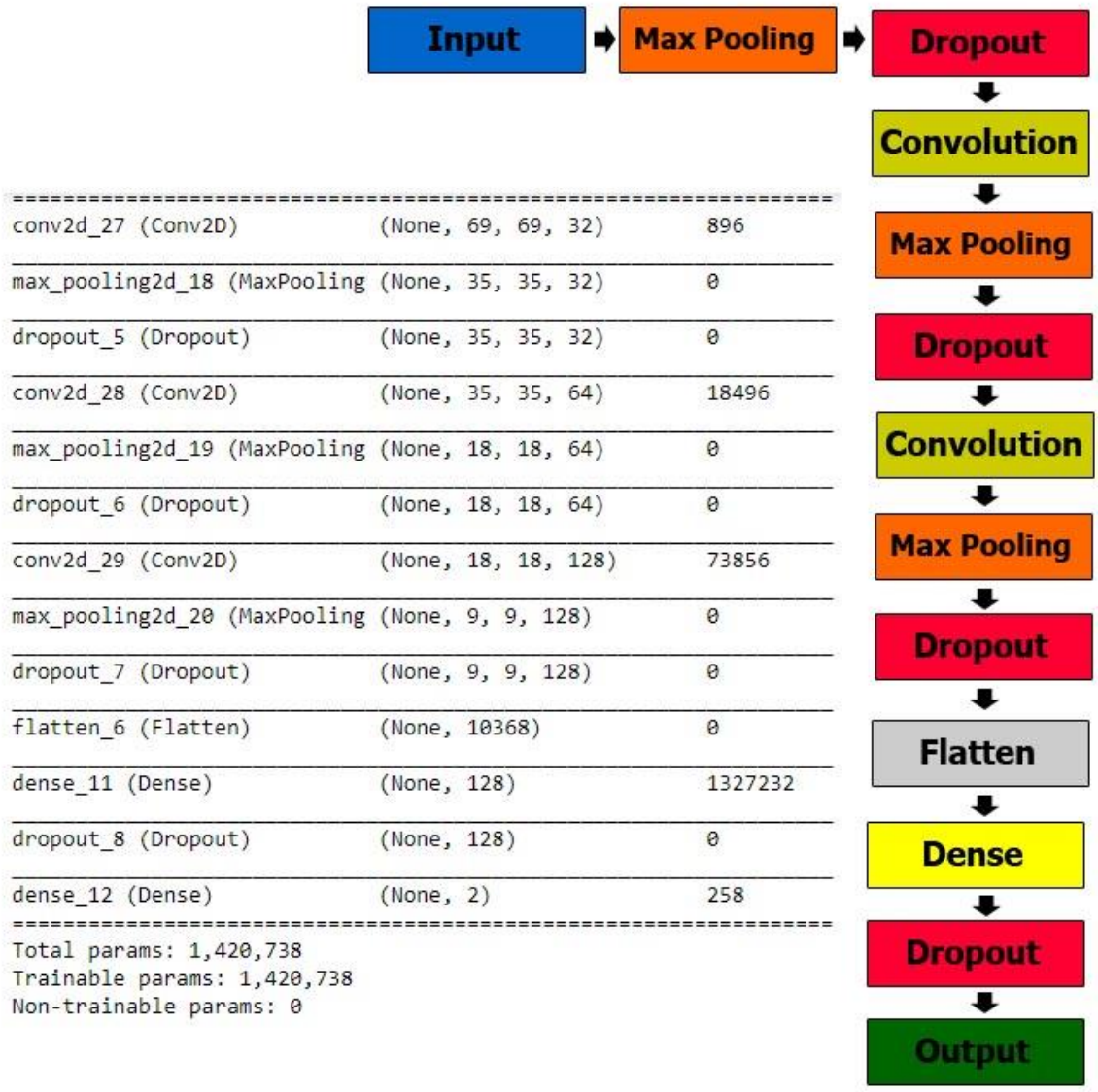




## 3.3. Refinement

After training and evaluating our initial and by observing the training accuracy and loss, I could say that the model did a good job since after 50 epochs the training accuracy is 93.13 % and the training loss is quite low. However, it looks like the model is **Overfitting**, gives an intuition that the network has memorized the training data very well but is not guaranteed to work on unseen data, and that is why there is a difference in the training and validation accuracy. So I have tried some optimization techniques such as trying to add more hidden layers of Convolution Filters, changing the size of window of MaxPool layer to get enhanced Feature Extraction, performing data augmentation, Regularization by putting **dropout** layers to avoid overfitting and other techniques until the model generalizes better and provides the desired output that works well for the astronomical images binary classification into real objects and Non-Real Objects task. Also I have changed the

optimization algorithm by using **Adam** optimizer and increasing both **number of epochs to 200** and **batch size to 128**.

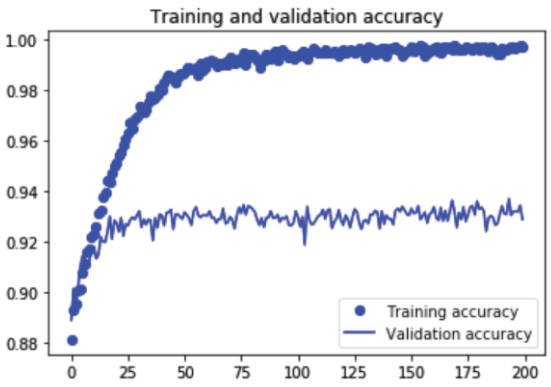


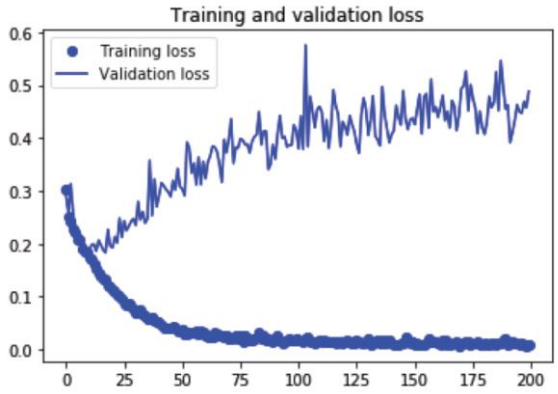
# 4. Results

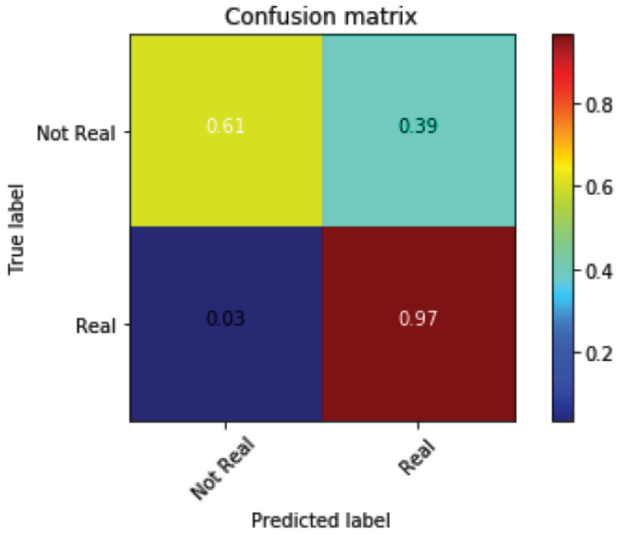
## 4.1. Model evaluation and validation

I have used the training set to tune the model parameters where the model learns from training data. But, in order to produce an independent measure of the model's performance a test set was used. Here I will introduce the final test and evaluation results including the Receiver Operating Characteristics (ROC) of our classifier based on its performance on final to compare to prior work.

| Accuracy | Recall | Precession | $F_1$-score |
|---|---|---|---|
| 92.36 % | 96.51 % | 94.93 % | 95.71 % |




As shown above, we tried to reduce overfitting. we obtained a suitable and acceptable accuracy on the test data substantially. The model with the **Dropout** layers performs the best on the test data.

### 4.2. Justification

In our solution as in the benchmark model, we considered the Accuracy and Recall to be the most important performance metrics since the two main classes in our data (Real objects and Non-real objects) are of similar sizes, **Accuracy** serves as a good general measure of performance, while **Recall** was chosen because we are more concerned with missing true objects (false negatives) than we are with contaminating our set of predicted objects with false positives since these are easy to human to exclude. These results suggest that it does in fact have a positive role to play in future astronomical surveys. Deep learning methods prove to be perfect player in this field, if not better, than human scanners; but unlike astronomers they can classify thousands of transients in a second. Unlike conventional machine learning algorithms convolutional neural networks require no complex and case-specific features to be crafted. With only data-augmentation during training, convolutional neural networks discover their own abstract features for classification. Deep models can provide continuous-valued scores for classification certainty that may be tuned, unlike human scanners, for best recall and precision. In addition, they are capable of handling such large data throughputs as may be generated by the Large Synoptic Survey Telescope through heavy utilization of GPUs. Deep models, in particular Convolutional Neural Networks, are indispensable for future astronomical sky surveys such as the LSST.

# 5. Conclusion

The benchmark model we are comparing with is examining a variety of machine learning algorithms for transient classification was simplified to perform the primary process of binary classification for the objects in the astronomical imaging data into Real and Non-Real objects. From the bench mark results we note that the best classifier was used is Random Forest (RF) with Accuracy and Recall at 91%. As well as it uses Neural Network called SkyNet that gives Accuracy of 88% and Recall of 89 %. But our final result using Deep Learning Convolutional Network (CNN) gives a good acceptable accuracy of 92.36 % and Recall 96.51 %, I have tried to get optimal solution for the problem and showing that using machine learning algorithms especially deep learning could exceed human manual scanning process and I think I have reached this goal with some degree.

# 6. Future works

Regarding future work using, my solution model working now on binary classification but I think as improvement, it should be upgraded to perform multi-classification of different categories in the imaging data to be more practical and useful tool for classification of supernovae and other objects in astronomical imaging data. As well as inject my future model with truly massive data sets considering the training to be done on GPUs. Also preprocessing algorithms may be developed for handling Signal-to-noise ratio and preparing difference images and centering candidate transients. In addition, I may include a time-series of images of the region in question, including time durations between scans images as features.