Image Classification Using Machine Learning and Deep Learning Model

K R Sinchana and Suman R Kunte

May 29, 2022

Image Classification Using Machine Learning And Deep Learning Model

Sinchana K R
Department Of Computer Science and Engineering
sinchanakr1207@gmail.com
Manipal Institute of Technology,Manipal,Udupi,Karnataka,India.

Suman R Kunte
Department Of Computer Science and Engineering
kuntesumanrk@gmail.com
Manipal Institute of Technology,Manipal,Udupi,Karnataka,India.

Abstract

A deep neural network (DNN) is an artificial neural network (ANN) with multiple layers between the input and output layers. Deeper neural networks are more difficult to train Many problems in computer vision were saturating on their accuracy before a decade. However, with the rise of deep learning techniques, the accuracy of these problems drastically improved. One of the major problems was that of image classification, which is defined as predicting the class of the image. Cat and Dog image classification is one such example of where the images of cat and dog are classified. This paper aims to incorporate state-of-art technique for object detection with the goal of achieving high accuracy. A convolutional neural network has been built for the image classification task.

Keywords

Image, Dog ,Cat, Classification, Relu, SoftMax, Pooling, Machine Learning.

Introduction

This article explores a Machine Learning algorithm called Convolution Neural Network (CNN), it's a common Deep Learning technique used for image recognition and classification. You are provided with a dataset consisting of 5,000 Cat images and 5,000 Dog images. We are going to train a Machine Learning model to learn differences between the two categories. The model will predict if a new unseen image is a Cat or Dog. The code architecture is robust and can be used to recognize any number of image categories, if provided with enough data. To learn about thousands of objects from millions of images, we need a model with a large learning capacity. However, the immense complexity of the object recognition task means that this problem cannot be specified even by a dataset as large as ImageNet, so our model should also have lots of prior knowledge to compensate for all the data we do not have. Thus, compared to standard feedforward neural networks with similarly sized layers, CNNs have much fewer connections and parameters and so they are easier to train, while their theoretically best performance is likely to be only slightly worse. Recently, deep ConvNets have significantly improved image classification and object detection accuracy. Compared to image classification, object detection is a more challenging task that requires more complex methods to solve. Due to this complexity, current approaches train models in multi-stage pipelines that are slow and inelegant. Complexity arises because detection requires the accurate localization of objects, creating two primary challenges.

The Dogs vs. Cats image classification has been around for a long time now. The Dogs vs. Cats competition from Kaggle is trying to solve the CAPTCHA challenge, which relies on the problem of distinguishing images of dogs and cats. It is easy for humans, but evidence suggests that cats and dogs are particularly difficult to tell apart automatically. Many people have worked or are working on constructing machine learning classifiers to address this problem. In my project I am going to build a convolutional neural network to solve the problem and achieve higher performance and better results. In my project instead of using the Kaggle data set comprising of total 25000 images, I would be working on subset of these images. My dataset would be comprising of total 10000 images. Kera's would use for model building and all the code would be implemented on google colab.

Dataset

The dataset in divided into folders for each class. The dataset is divided into training dataset and testing dataset. The training dataset and the testing dataset compose of 2 folders one for cat images and other for dog images. There are 10000 images in total. Where each cat and dog image in testing dataset contain 8000 image and in training contains 2000. The images are of varying shape and sizes, but in order to train a CNN the images should be of same size. Data Augmentation is being carried out by using the ImageDataGenerator module provided by Kera's. Using it the images are resized to 64 x 64. Also, for training of a convolutional neural network large set of images are needed. So, data augmentation is applied on the existing set of images to increase the dataset size. Various data augmentation technique such as rescaling, shear range, zoom range are being used to do the same.

Methods

Convolution Neural Networks (CNN)

Convolution Neural Networks are good for pattern recognition and feature detection which is especially useful in image classification. Improve the performance of Convolution Neural Networks through hyper-parameter tuning, adding more convolution layers, adding more fully connected layers, or providing more correctly labelled data to the algorithm.

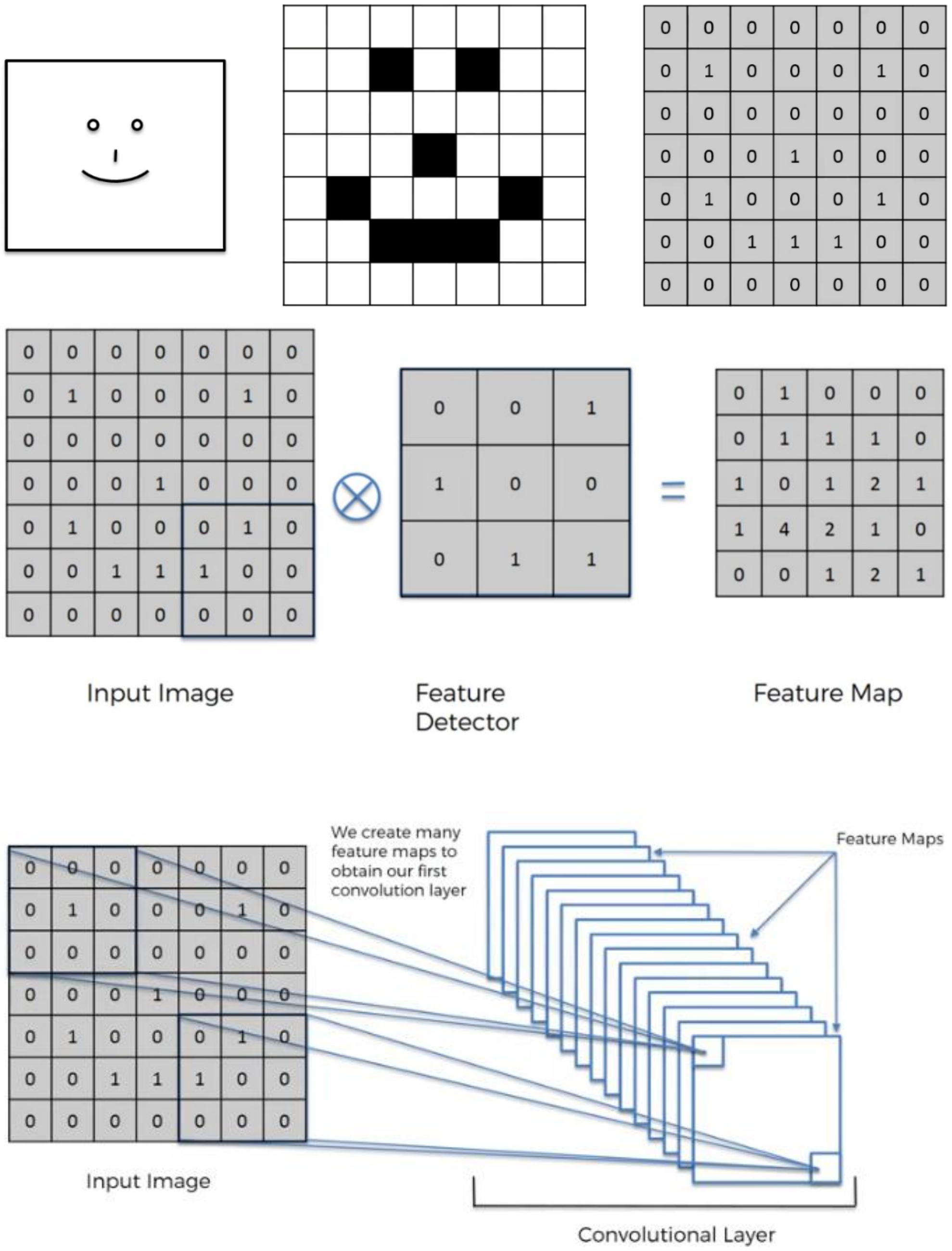
Create a Convolution Neural Network (CNN) with the following steps:

1. Convolution
2. Max Pooling
3. Flattening
4. Full Connection

Convolution is a function derived from two other functions through an integration that expresses how the shape of one is modified by the other.

$$(f * g)(t) \overset{\text{def}}{=} \int_{-\infty}^{\infty} f(\tau)\, g(t-\tau)\, d\tau$$

For image recognition, we convolve the input image with Feature Detectors (also known as Kernel or Filter) to generate a Feature Map (also known as Convolved Map or Activation Map). This reveals and preserves patterns in the image, and also compresses the image for easier processing. Feature Maps are generated by element-wise multiplication and addition of corresponding images with Filters consisting of multiple Feature Detectors. This allows the creation of multiple Feature Maps.



0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	1	0	0	0	1	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0

Input Image

0	0	1
1	0	0
0	1	1

Feature Detector

0	1	0	0	0
0	1	1	1	0
1	0	1	2	1
1	4	2	1	0
0	0	1	2	1

Feature Map

We create many feature maps to obtain our first convolution layer
Feature Maps
Input Image
Convolutional Layer

Fig 1:Convolution Operataion

This Image Convolution allows you to play with various filters applied to an image. Edge Detect is a useful filter in Machine Learning. The algorithm creates filters that are not recognizable to humans, perhaps we learn with similar techniques in our subconscious. Feature Maps preserve spatial relationships between pixels throughout processing.

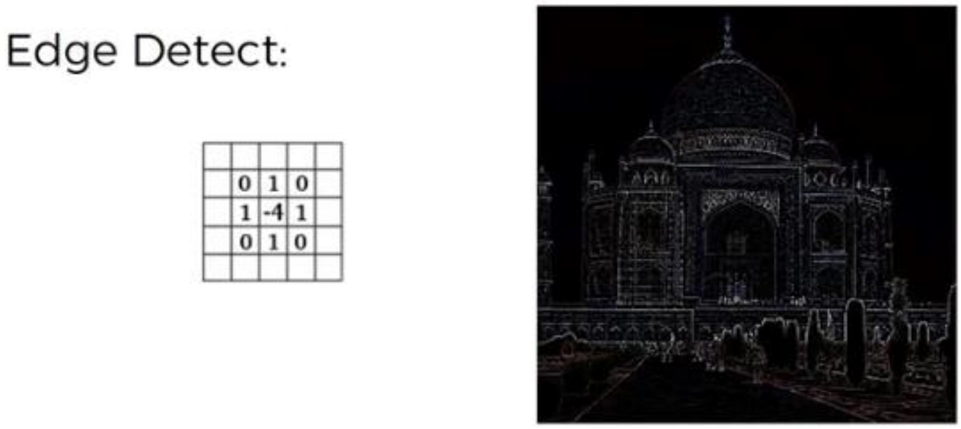


Fig 2:Edge Detection

Rectified Linear Units (ReLU)

Rectifier Functions are applied to Convolution Neural Networks to increases non-linearity (breaks up linearity). This is an important step for image recognition with CNNs. Images are usually non-linear due to sharp transition of pixels, different colours, etc. ReLU functions help amplify the non-linearity of images, so the ML model has an easier time finding patterns.

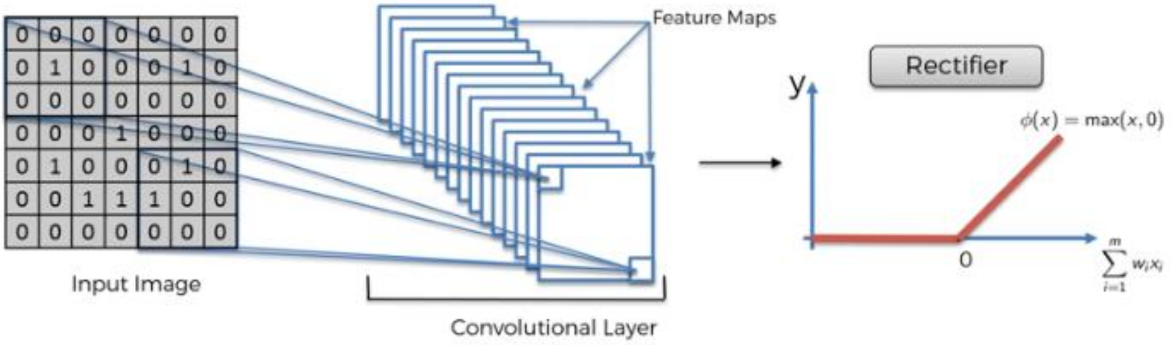
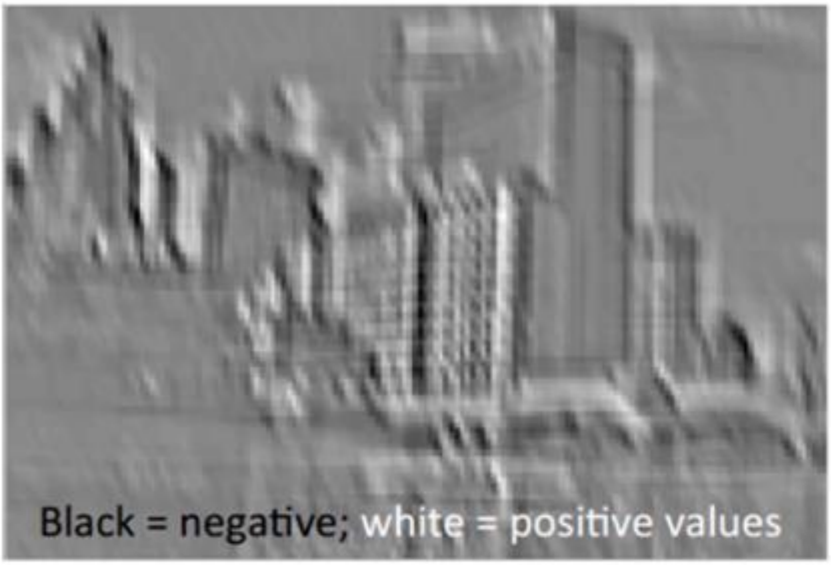


Fig 3:ReLU activation function

Before ReLU



After ReLU

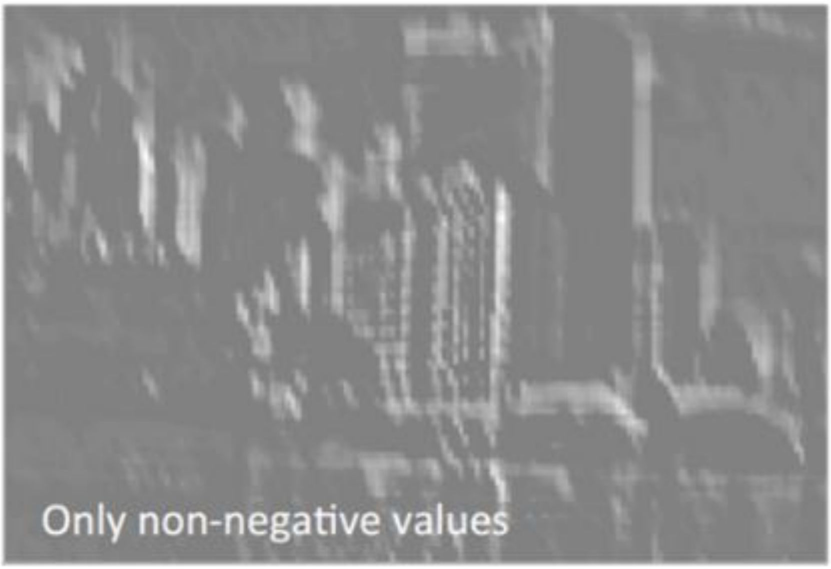
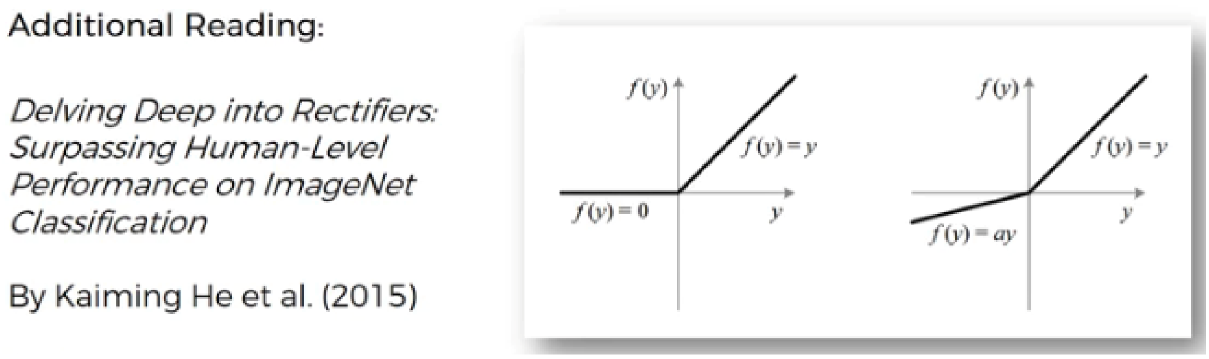


Fig 4 and 5: Before and After ReLU

In the above example, the ReLU operation removed the Black Pixels so there's less White to Gray to Black transitions. Borders now have more abrupt Pixel changes. Microsoft argues that the using their Modified Rectifier Function works better for CNNs.



Additional Reading:

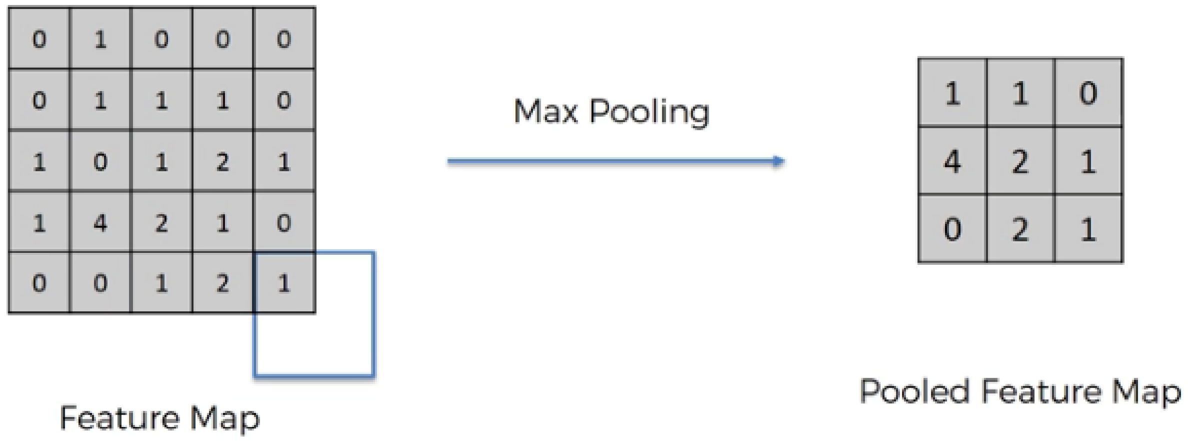
Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification

By Kaiming He et al. (2015)

Fig 6: Activation Representation

Max Pooling

Max Pooling is a pooling operation that calculates the maximum value for patches of a feature map and uses it to create a down sampled (pooled) feature map. It is usually used after a convolutional layer. It adds a small amount of translation invariance - meaning translating the image by a small amount does not significantly affect the values of most pooled outputs.



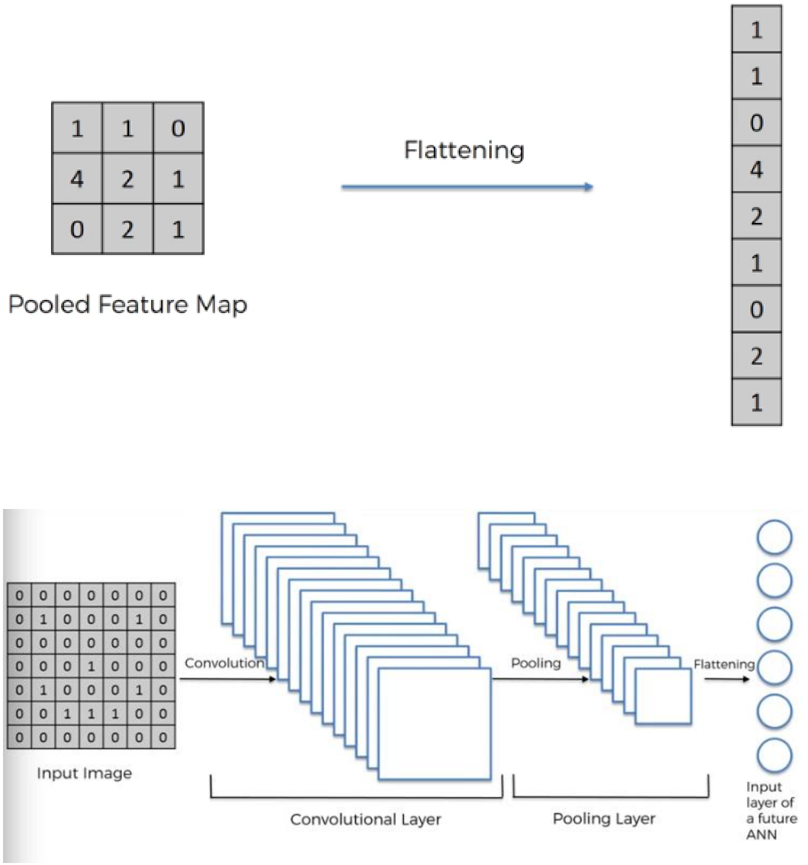
0	1	0	0	0
0	1	1	1	0
1	0	1	2	1
1	4	2	1	0
0	0	1	2	1

1	1	0
4	2	1
0	2	1

Fig 7: Max Pooling

Flattening

Flattening puts values of the pooled Feature Map matrix into a 1-D vector. This makes it easy for the image data to pass through an Artificial Neural Network algorithm.



1	1	0
4	2	1
0	2	1

1
1
0
4
2
1
0
2
1

0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	1	0	0	0	1	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0

Fig 8: Flattening

Full Connection

This is when the output of a Convolution Neural Network is flattened and fed through a classic Artificial Neural Network. It's important to note that CNNs require fully connected hidden layers whereas regular ANNs don't necessarily need full connections.

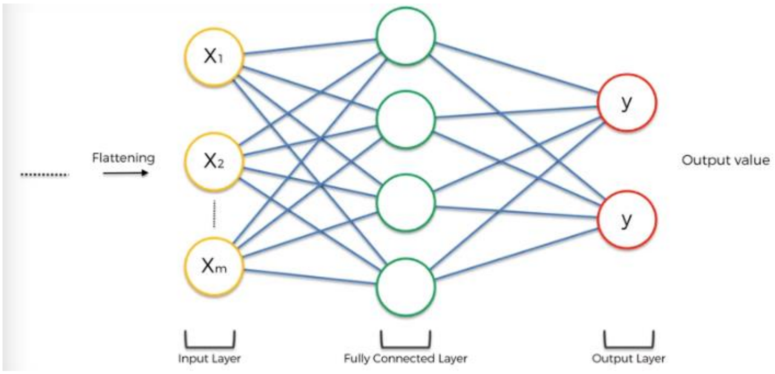


Fig 9 : CNN Representation

The process of CNN back-propagation adjusts weights of neurons, while adjusting Feature Maps.

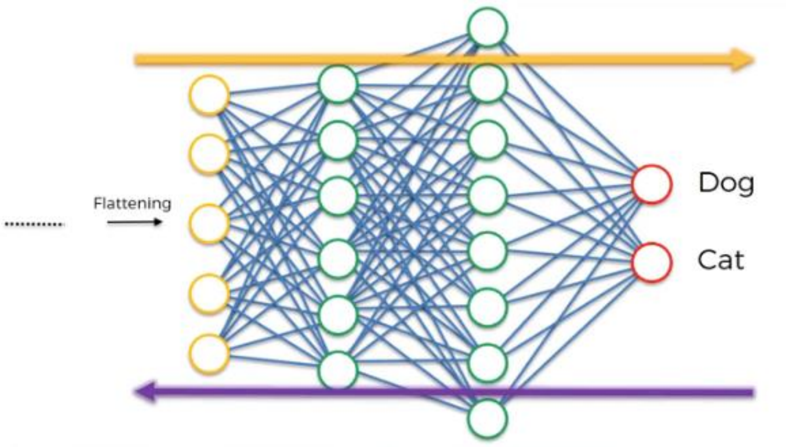


Fig 10:CNN classification for Cat v/s Dog

When it's time for the CNN to make a decision between Cat or Dog, the final layer neurons 'vote' on probability of an image being a Cat or Dog (or any other categories you show it). The Neural Network adjusts votes according to the best weights it has determined through back-propagation.

The depiction of the CNN architecture used in this work is presented . The system comprises of three-layered architecture (two convolutional-pooling and one completely connected), aside from last layer of output neurons (excluding the input layer). The parameters of the CNN are chosen on the basis of accuracy and loss. The input consists of 64×64×3 neurons, signifying the RGB values for a 64×64×3 picture. The primary convolutional-pooling layer utilizes a local receptive field (otherwise called convolutional kernel). The kernel has a size of 3 × 3 and stride length of 1 pixel to separate 32 feature maps, trailed by a max pooling task directed in a 2×2 region. The next convolutional-pooling layer additionally utilize 3×3 local receptive fields, bringing about 32 feature maps, and the remaining parameters stay unaltered. The third layer consists of 128 rectified linear units (ReLU) neurons which is a completely connected layer, and the last layer has 2(binary) neurons that relate to the classification of cat or dog. The ReLU activation function is utilized by the two convolutional-pooling layers. ReLU lessens the gradient vanishing issue in CNN by spreading gradient proficiently

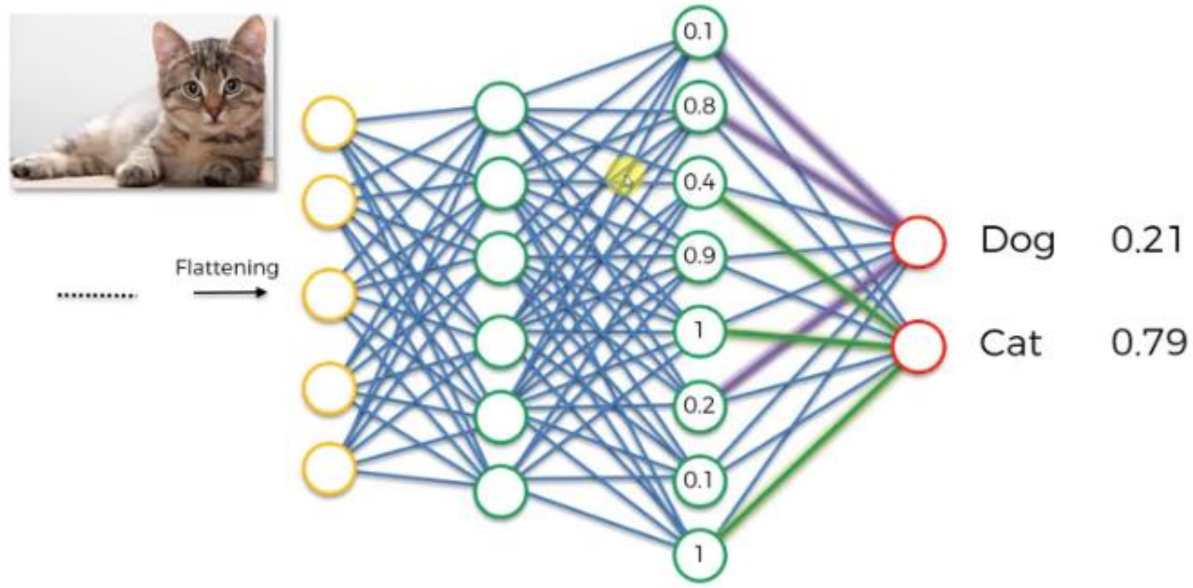
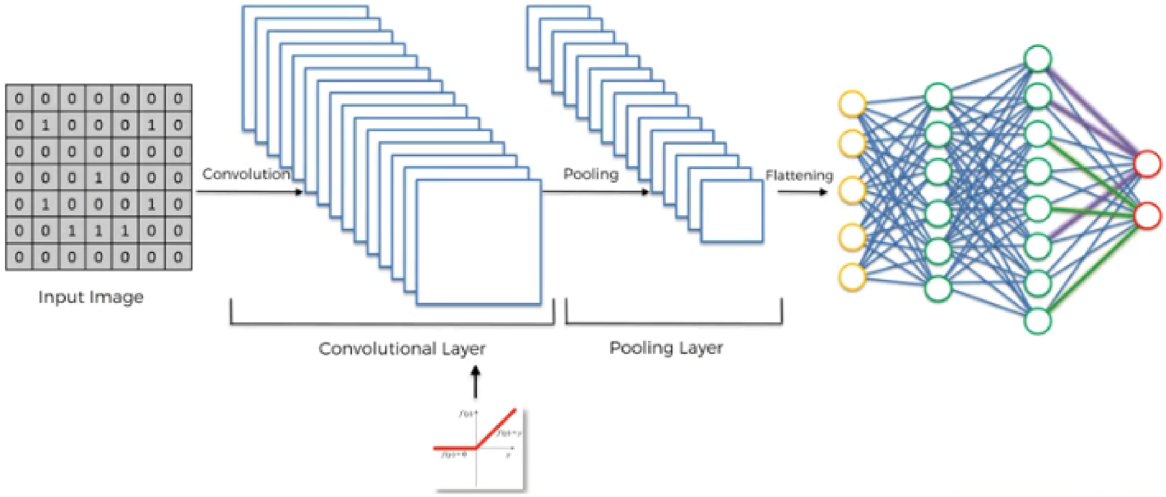


Fig 11: Image Flattening

Here is a summary of every step of a CNN, don't forget about the Rectifier Function that removes linearity in Feature Maps, also remember that the hidden layers are fully connected.



0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	1	0	0	0	1	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0

Fig 12: CNN using ReLU activation function

Pre-Processing (Images Augmentation)

This step modifies images to prevent over-fitting. This data augmentation trick can generate tons more data by applying random modifications to existing data like shearing, stretching, zooming, etc. This makes your dataset and algorithm more robust and generalized.

SoftMax and Cross-Entropy Cost Function

When working on a Machine Learning or a Deep Learning Problem, loss/cost functions are used to optimize the model during training. The objective is almost always to minimize the loss function. The lower the loss the better the model. Cross-Entropy loss is a most important cost function. It is used to optimize classification models. The understanding of Cross-Entropy is pegged on understanding of SoftMax activation function

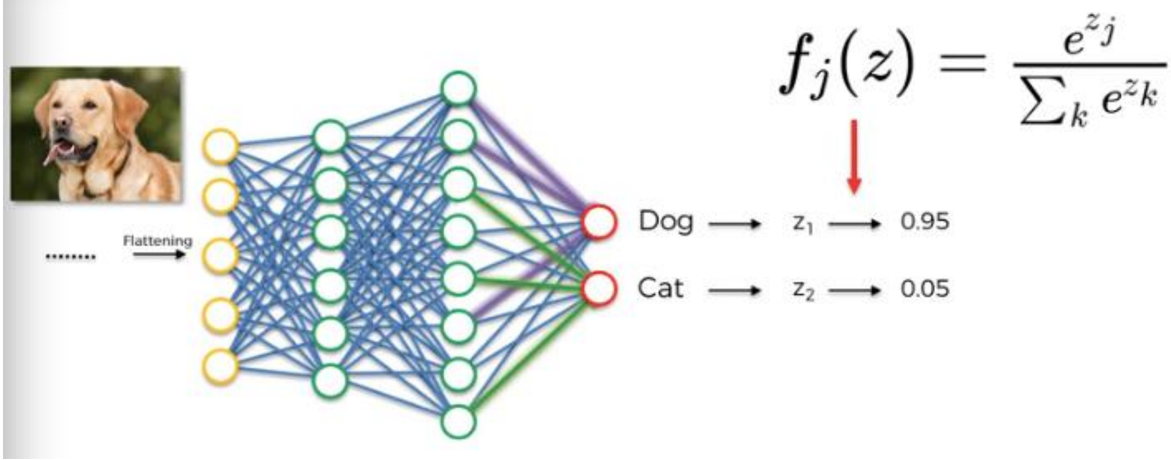
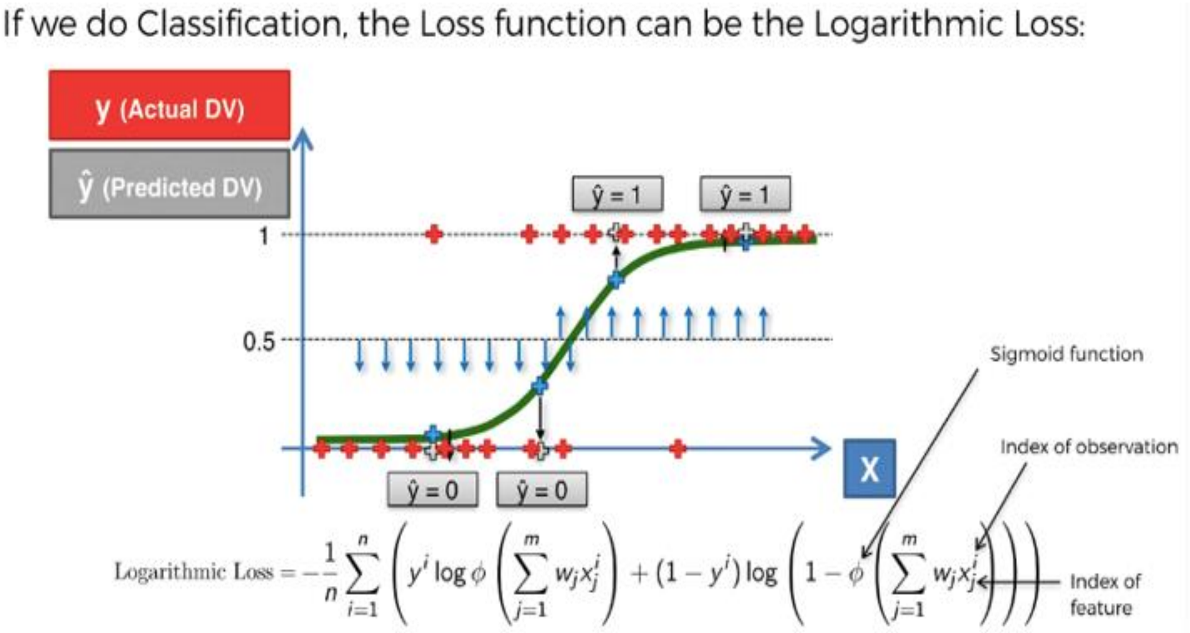


Fig 13: Dog image Classification

We had previously used the Mean Squared Error (MSE) Cost Function. For CNNs, it's better to use the Cross-Entropy Function as your Cost Function. We use Cross-Entropy as a Loss Function because it has a 'Log' term which helps amplify small Errors and better guide gradient descent.

Logarithmic Loss

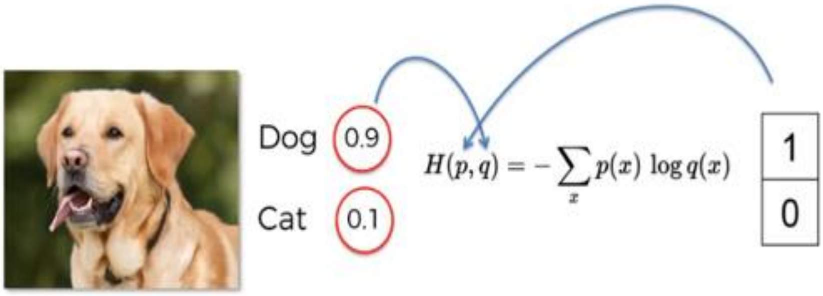


If we do Classification, the Loss function can be the Logarithmic Loss:

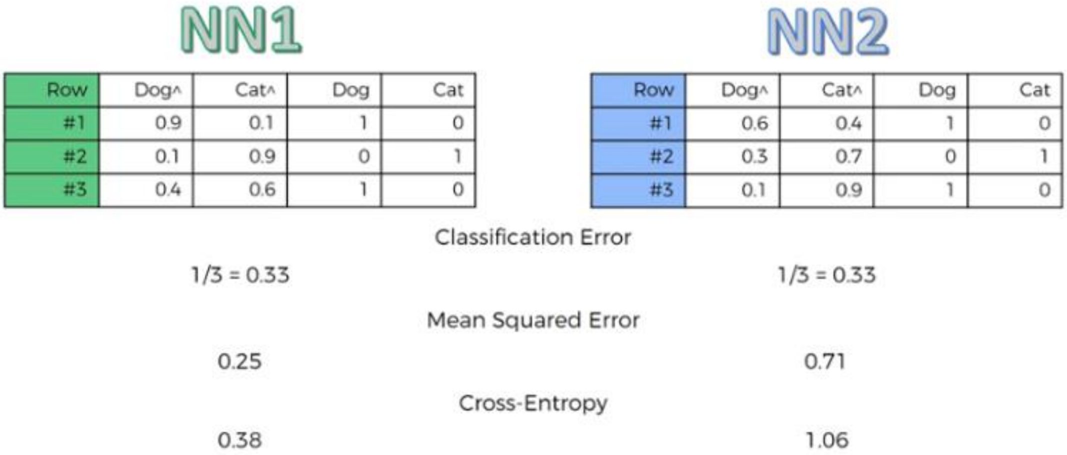
$$\text{Logarithmic Loss} = -\frac{1}{n}\sum_{i=1}^{n}\left(y^i \log\phi\left(\sum_{j=1}^{m} w_j x_j^i\right) + (1-y^i)\log\left(1-\phi\left(\sum_{j=1}^{m} w_j x_j^i\right)\right)\right)$$

$$L_i = -\log\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right)$$

$$H(p,q) = -\sum_x p(x)\,\log q(x)$$



$$H(p,q) = -\sum_x p(x)\,\log q(x)$$



NN1

Row	Dog^	Cat^	Dog	Cat
#1	0.9	0.1	1	0
#2	0.1	0.9	0	1
#3	0.4	0.6	1	0

NN2

Row	Dog^	Cat^	Dog	Cat
#1	0.6	0.4	1	0
#2	0.3	0.7	0	1
#3	0.1	0.9	1	0

Classification Error

1/3 = 0.33 1/3 = 0.33

Mean Squared Error

0.25 0.71

Cross-Entropy

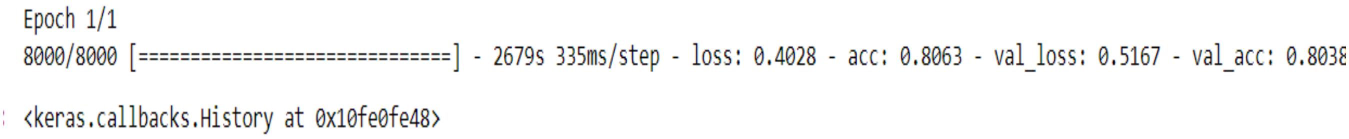
0.38 1.06

Fig 14 :Representation of Logarithmic Loss

Results

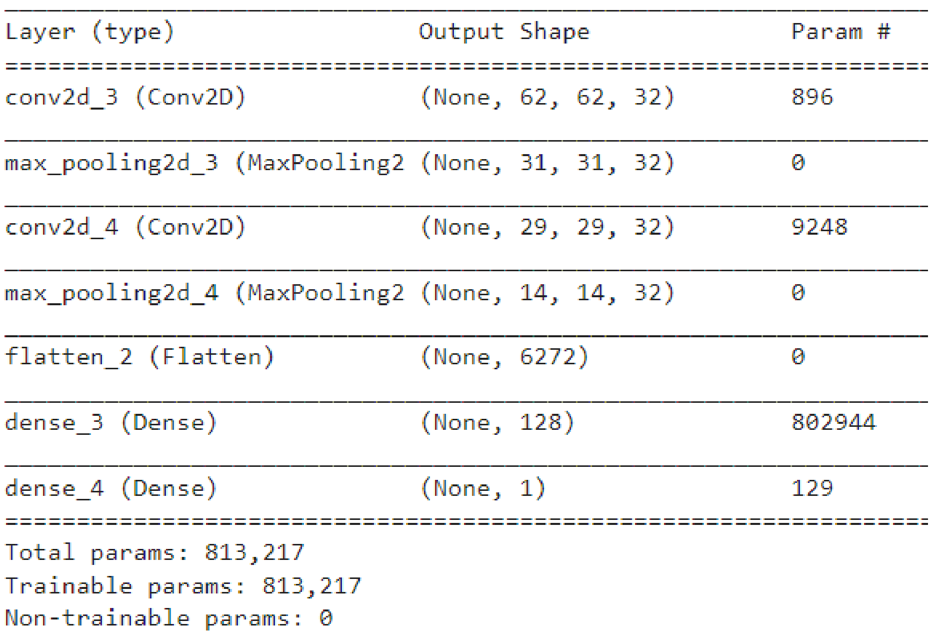
There are several tests conducted from this trained model. To prove trained model works, prediction done in computer which used in training using new test images Result model with lowest loss in validation is used (0.55) with accuracy 0.78 in validation images.

There are several tests conducted from this trained model. To prove trained model works, prediction done in computer which used in training using new test images

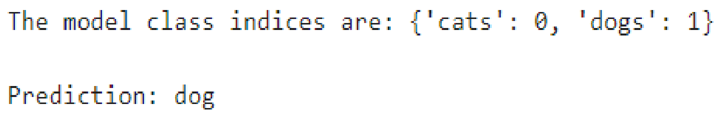


```
Epoch 1/1
8000/8000 [==============================] - 2679s 335ms/step - loss: 0.4028 - acc: 0.8063 - val_loss: 0.5167 - val_acc: 0.8038

<keras.callbacks.History at 0x10fe0fe48>
```



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_3 (Conv2D)            (None, 62, 62, 32)        896       
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 31, 31, 32)        0         
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 29, 29, 32)        9248      
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 14, 14, 32)        0         
_________________________________________________________________
flatten_2 (Flatten)          (None, 6272)              0         
_________________________________________________________________
dense_3 (Dense)              (None, 128)               802944    
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 129       
=================================================================
Total params: 813,217
Trainable params: 813,217
Non-trainable params: 0
```



```
The model class indices are: {'cats': 0, 'dogs': 1}

Prediction: dog
```

Fig 15: Result of Prediction of Dog V/S Cat Image

Conclusion

This study examined the efficacy of the pre training Convolutional neural network with natural images of cat -dog dataset which has around 4000 training images for each of the category. We achieved the best classification accuracy of 88.31 % with the optimal parameter settings of the proposed architecture. Training the CNN with various epochs and with restricted layers demonstrated the constrained space for further change in test accuracy. Gathering additional training data or by streamlining the design (including more layers) and hyperparameters of the system the test accuracy can be significantly improved.

References

1. Jajodia, T. and Garg, P., 2019. Image classification–cat and dog images. *Image*, *6*(23), pp.570-572
2. Kaggle DogVCat Competation: http://www.kaggle.com/c/dogs-vs-cats
3. S. Panigrahi, A. Nanda and T. Swarnkar, "Deep Learning Approach for Image Classification," 2018 2nd International Conference on Data Science and Business Analytics (ICDSBA), 2018, pp. 511-516, doi: 10.1109/ICDSBA.2018.00101.
4. H. A. Shiddieqy, F. I. Hariadi and T. Adiono, "Implementation of deep-learning based image classification on single board computer," 2017 International Symposium on Electronics and Smart Devices (ISESD), 2017, pp. 133-137, doi: 10.1109/ISESD.2017.8253319.
5. L Perez, J Wang - arXiv preprint arXiv:1712.04621, 2017 - arxiv.org
6. Neha Sharma, Vibhor Jain, Anju Mishra,An Analysis Of Convolutional Neural Networks For Image Classification,Procedia Computer Science,Volume 132,2018,Pages 377-384,ISSN 1877-0509,
https://doi.org/10.1016/j.procs.2018.05.198.
7. Tamuly, S., Jyotsna, C., Amudha, J. (2020). Deep Learning Model for Image Classification. In: Smys, S., Tavares, J., Balas, V., Iliyasu, A. (eds) Computational Vision and Bio-Inspired Computing. ICCVBIC 2019. Advances in Intelligent

Systems and Computing, vol 1108. Springer, Cham. https://doi.org/10.1007/978-3-030-37218-7_36

8. Shu, Mengying, "Deep learning for image classification on very small datasets using transfer learning"
(2019). Creative Components. 345.
https://lib.dr.iastate.edu/creativecomponents/345
9. R. Kumar, M. Sharma, K. Dhawale and G. Singal, "Identification of Dog Breeds Using Deep Learning," 2019 IEEE 9th International Conference on Advanced Computing (IACC), 2019, pp. 193-198, doi: 10.1109/IACC48062.2019.8971604.
10. F. Sultana, A. Sufian and P. Dutta, "Advancements in Image Classification using Convolutional Neural Network," 2018 Fourth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN), 2018, pp. 122-129, doi: 10.1109/ICRCICN.2018.8718718.\
11. [09:45] SUMAN R KUNTE - 210913026
12. Z. Hu, Y. Li and Z. Yang, "Improving Convolutional Neural Network Using Pseudo Derivative ReLU," 2018 5th International Conference on Systems and Informatics (ICSAI), 2018, pp. 283-287, doi: 10.1109/ICSAI.2018.8599372.
13. Sharma, N., Jain, V., & Mishra, A. (2018). "An Analysis of Convolutional Neural Networks for Image Classification", Procedia Computer Science, Vol.132, pp.377-384.
14. M. A. Arefeen, S. T. Nimi, M. Y. S. Uddin and Z. Li, "A Lightweight Relu-Based Feature Fusion for Aerial Scene Classification," 2021 IEEE International Conference on Image Processing (ICIP), 2021, pp. 3857-3861, doi: 10.1109/ICIP42928.2021.9506524.
15. S.H. Shabbeer Basha, Shiv Ram Dubey, Viswanath Pulabaigari, Snehasis Mukherjee,Impact of fully connected layers on performance of convolutional neural networks for image classification,Neurocomputing,Volume 378,2020,Pages 112-119,ISSN 0925-2312,https://doi.org/10.1016/j.neucom.2019.10.008.(https://www.sciencedirect.com/science/article/pii/S0925231219313803)
16. S. Liu and W. Deng, "Very deep convolutional neural network-based image classification using small training sample size," 2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR), 2015, pp. 730-734, doi: 10.1109/ACPR.2015.7486599.
17. K. K. Pal and K. S. Sudeep, "Preprocessing for image classification by convolutional neural networks," 2016 IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT), 2016, pp. 1778-1781, doi: 10.1109/RTEICT.2016.7808140.