Security application for public places using CNN for Image Classification

Research Paper

Kshitij Sinha*, Muskan Lalit*

Kshitij Sinha - Indian Institute of Information Technology, Bhopal, India
Muskan Lalit - Indira Gandhi Delhi Technical University for Women, Delhi, India
Created 26th March 2020

Abstract

The concept of Neural Networks in the realm of Deep Learning has undeniably been one of the most notable breakthroughs in Machine Learning and AI. In this paper, we elaborate and further apply the concepts of Deep Learning algorithms namely Convolutional Neural Networks(CNN) in image classification. We have curated a workable dataset of 4000 images of toddlers and obtained the dataset of pets(dogs/cats) from Kaggle[1]. We used CNN to train our model over several epochs to obtain an accurate result whilst avoiding overfitting. The model was evaluated using a classification accuracy and loss plot. The purpose of this research is to increase automation in public places such as malls, movie theatres, and similar places which host mass gatherings of people and shouldn't include pets and other animals. Our aim through this project is to develop an intelligent application/software that can detect objects through images captured by security cameras and then classify them as permitted or not. Our project achieved this by the use of CNN for object classification and our vision is to develop an application/software which is assisted by our model to permit or block the entrance of undesired objects/animals.

Keywords: CNN, Object Classification, Application of CNN, Security, Deep Learning, Machine Learning

1. Introduction

Image processing involves some basic operations such as image restoration/rectification, image enhancement, image classification, image fusion, etc. Image classification forms an important part of image processing. The objective of image classification is the automatic allocation of image to thematic classes . Two types of classification are supervised classification and unsupervised classification. This project is based on supervised binary image classification, the two classes being toddlers and pets. Through this project we aim to develop an intelligent model that can aid in the security systems for public places by detecting and classifying images taken through the security cameras into the two mentioned categories and labelling them as permissible or not. We employ the CNN architecture for the training and testing of our dataset and further evaluate and plot classification accuracy to validate our results.

In this paper, we elaborate upon the working of convolutional neural networks(CNN) and it's related concepts that include error function, loss function, hyperparameter tuning, backpropagation and gradient descent. This explanation is in coherence with our project and each proposition is facilitated with code fragments for the same. We conclude this paper by presenting our results that substantiate our model.

2. Dataset

We procured the dataset for pets from Kaggle[1]. containing 4500 images. For creating the dataset of toddler images we used the chrome extension - 'Download All Images'. We downloaded an approximate number of 4300 images for the same. For both the classes we divided the dataset into a ratio of 4:1 (approx.) for training and testing respectively. This dataset was divided into 96 batches and trained over 10 epochs(as a demonstration) which helped achieve great results for our CNN model.

3. Working of the Network

Supervised vs. Unsupervised Learning

Supervised Learning involves an independent set of features(input values(X)) and a dependent variable(output value(Y)). The goal is to learn a mapping function well enough that is able to predict the output values(Y) for any set of new given input values(X) accurately.

*The author names are in the alphabetical order of their first names and no other particular order

Unsupervised learning is a type of machine learning algorithm used to draw inferences from datasets consisting of input data without labeled responses.The most common unsupervised learning method is cluster analysis, which is used for exploratory data analysis to find hidden patterns or grouping in data. The clusters are modeled using a measure of similarity which is defined upon metrics such as Euclidean or probabilistic distance.

Classification

Classification is the problem of identifying to which of a set of categories (sub-populations) a new observation belongs, on the basis of a training set of data containing observations (or instances) whose category membership is known. It is a type of supervised learning and specifies the class to which data elements belong to. It is best used when the output has finite and discrete values.

Image classification is the process of taking an input image and outputting the probability that the input is from a particular class(such as “dog”). We use neural networks for these image classification problems.

Types of classifiers in a CNN

In CNN using keras we have two types of models under keras.models library:

1. Sequential Model

   The Sequential model is a linear stack of layers.The idea of sequential (multistage) classifiers is to break up a complex decision into a collection of several simpler decisions.

2. Model API

   The functional API in Keras is an alternate way of creating models that offers a lot more flexibility, including creating more complex models.It specifically allows you to define multiple input or output models as well as models that share layers.

This paper uses the Sequential classifier for its CNN model.

```
classifier = Sequential()
```

Neural Networks - A network of Neurons

The goal of neural networks is to make the Machine learn without human intervention, or simply, give the computer a brain of its own. And the most obvious way to do this was to try and replicate the working of the human brain to as much extent as possible.

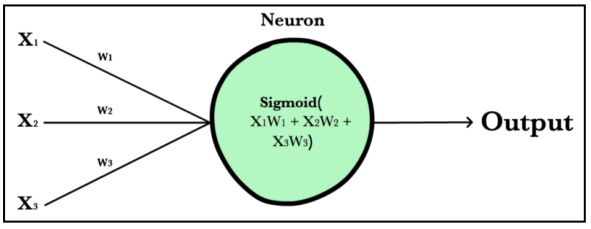

Fig3.1:Structure of a neuron and its working

X1, X2, X3 are the inputs of this neuron. The connecting lines or bridges, technically called Synapses are assigned some weights, say w1, w2, w3. A weighted input as the weights are multiplied with their associated inputs and then summed together.

The expression looks like this

$Z=\sum(i=1 \text{ to } n)\ xiwi=x1w1+x2w2+x3w3$ -(1)

It is able to give a quantitative importance to each of the weights relative to each other, or “extract features based on their importance”.A function is then applied to this expression for manipulation known as the Activation Function which gives the output as a function of all the inputs.

An additional parameter in the Neural Network which is used to adjust the output along with the weighted sum of the inputs to the neuron.

```
output  =  sum (weights * inputs) + bias
```

A number of neurons at one level or “layer” and a number of such layers all being connected to each other by synapses. This is known as a neural network. The output from one neuron in one layer, serves as one of the inputs for neurons in the next layer.

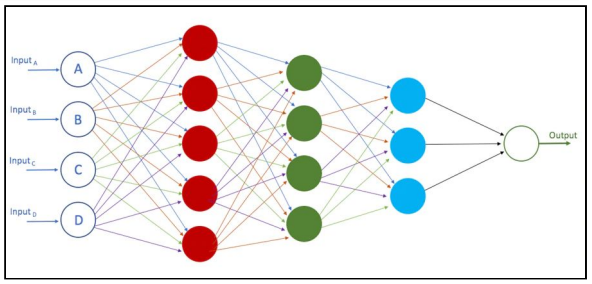

Fig3.2: A fully connected Neural Network with 4 inputs, 3 layers and 1 Output.

Activation Function

Activation Functions are able to perform linear as well as non-linear mappings between the inputs and the output.

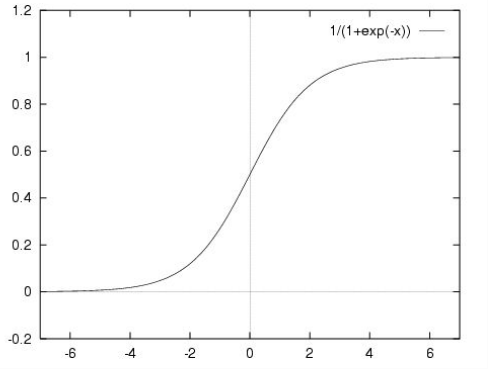

Fig3.3: Sigmoid activation function:transforms the input into a number between 0 and 1

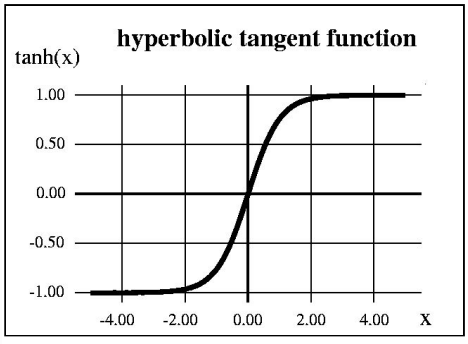

Fig3.4:Tanh activation function:transforms the input into a number between -1 and 1

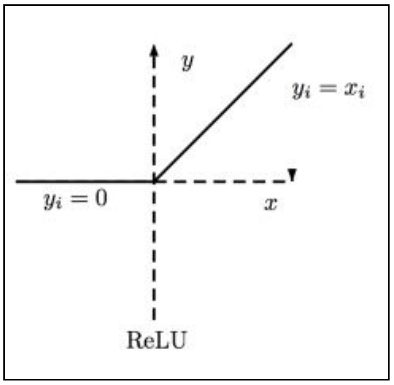

Fig3.5:ReLU: R(x) = max{0, x}

Convolutional Neural Network (CNN)

Convolutional neural network involves preprocessing(feature learning) of a given multidimensional input to extract a one dimensional vector of features which is fed as input to a fully connected neural network.

CNN involves the use of Convolution layers along with Pooling and Activation for feature extraction. It transfers the responsibility of learning the important features and extracting them to make a single vector on to the machine.

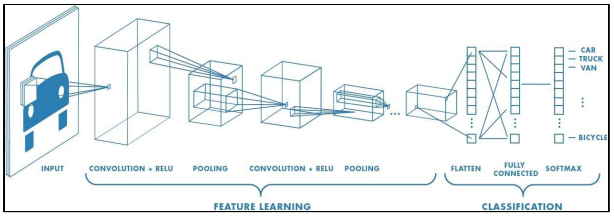

Fig3.6: CNN Architecture

Input

The input image is a 3D array of pixels. It consists of 3 “channels” which are the RGB channels (Red-Green-Blue). Each pixel is an 8-bit binary number or simply a decimal number between 0 and 255.

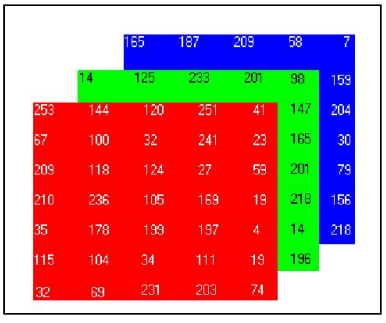

Fig 3.7: RGB channels of an image

Say, our image is of dimensions 256x256, then the computer can see a 3D array of dimensions 256x256x3. For ease of understanding this paper shall show all operations of CNN for a 2D array of dimensions 256x256. All the results can be generalized for 3D arrays as well.

Convolution Layer

The first layer of the CNN that processes the input is known as the Convolution Layer. The main component of this layer is a filter or kernel a.k.a. Feature Detector.

Feature Detector

Feature detectors are matrices of specific values and are used to highlight features in an input image. These feature detectors are also referred to as kernels or filters. There are various types of filters:

Blur image filter

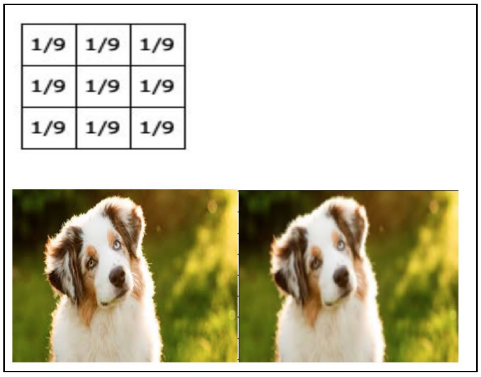

Fig 3.8:Filter for blurring an image

Sharpen image filter

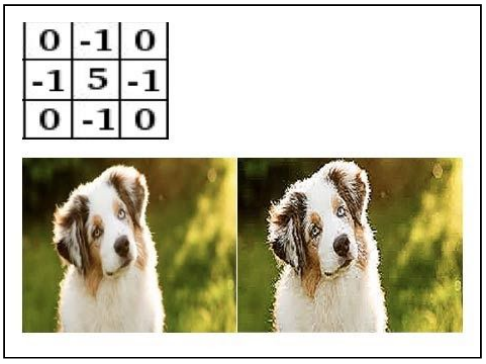

Fig 3.9:Filter for sharpening an image

Edge detection filter

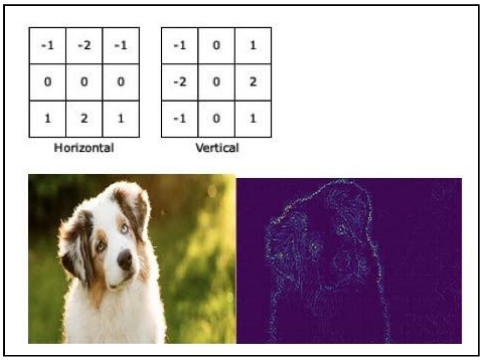

Fig3.10:Filter for detecting and highlighting edges

Convolution Operation

Feature detectors work in a very simple way to highlight all the features of an input image. The feature detectors translate over the image and the value in each element of the feature detector is multiplied by every corresponding element of the area of the input image covered (say if it's a 3x3 matrix over a 8x8 input image then a certain 3x3 shall be occupied at each step) and the sum of values is written as one element in the resulting matrix. The resulting matrix is called the feature map and is a matrix of reduced size.

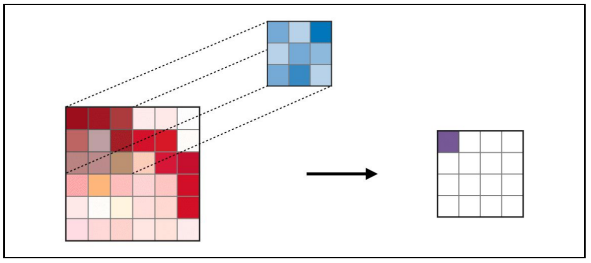

Fig 3.11: Working of a feature detector

Data loss occurs while we create feature maps as we are reducing the size of the image but on the other hand we are highlighting the features by using feature maps.

Activation Function Layer(ReLU Layer)

The Activation Function that is used to train deep networks must provide high sensitivity to the activation sum input and should not be easily saturated. A piecewise linear function i.e. the ReLU function works perfectly as it looks and acts like a linear function but is, in fact, a nonlinear function, thus allowing complex relationships to be learned. It also makes it possible to train our model using back propagation of errors using stochastic gradient descent algorithm.

```
classifier.add(Convolution2D(filters = 32, kernel_size = (3, 3), input_shape = (128, 128, 3), activation = 'relu'))
```

Pooling

The purpose of the pooling layer is to reduce the dimension of the matrix obtained in the ReLU layer but keep all the essential features. It is useful for extracting dominant features which are rotational and positional invariant, thus maintaining the process of effectively training the model.

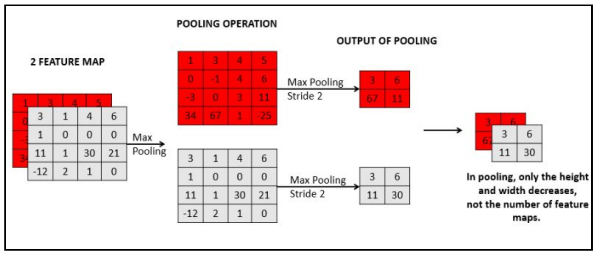

Fig 3.12: Pooling operation

This layer drastically reduces the spatial dimension (the length and the width change but not the depth) of the input volume. This serves two main purposes. The first is that the amount of parameters or weights is reduced by 75%, thus lessening the computation cost. The second is that it will control overfitting.

```
classifier.add(MaxPooling2D(pool_size = (2,2)))
```

Extracting High level features

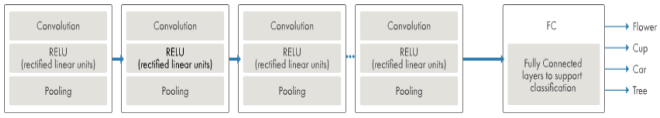

Fig 3.13: Architecture of CNN

The layers occur multiple times over for a single image to be able to extract complex features such as shapes and structures in the image.The first ConvLayer is responsible for capturing the Low-Level features such as edges, color, gradient orientation, etc. With added layers, the architecture adapts to the High-Level features as well, giving us a network which has a wholesome understanding of images in the dataset. Depending on the complexities in the images, the number of such layers may be increased for capturing low-levels details even further, but at the cost of more computational power.

```
classifier.add(Convolution2D(32, (3, 3), activation = 'relu'))
classifier.add(MaxPooling2D(pool_size = (2,2)))
```

This paper uses two layers of Convolution and Pooling for better feature extraction.

Flattening

This layer takes the matrices obtained in the pooled feature map and transforms them into a single 1D vector of features. This vector can then be fed to the neural network for training and testing.

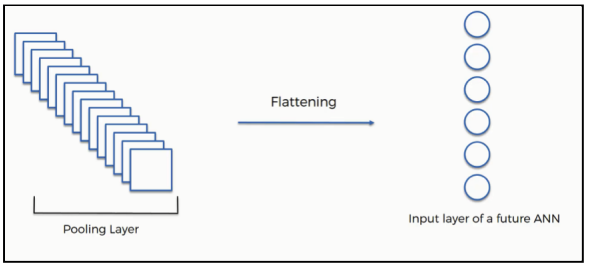

Fig 3.14: Flattening operation

```
classifier.add(Flatten())
```

Fully Connected Layers

Fully Connected Layers form the last few layers in the network. The input to the fully connected layer is the output from the final Pooling or Convolutional Layer, which is flattened and then fed into the fully connected layer. This Flattened vector is then connected to a few fully connected layers which are the same as Artificial Neural Networks and perform the same mathematical operations.

```
classifier.add(Dense(activation = 'relu', units = 128))
```

```
classifier.add(Dense(activation = 'sigmoid', units = 1))
```

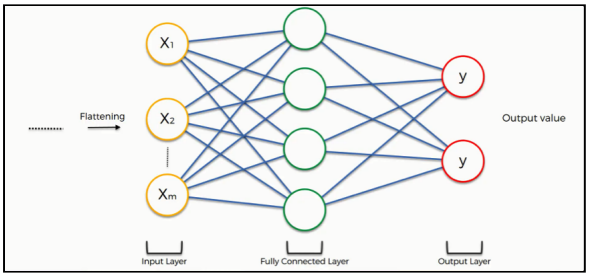

Fig 3.15: Working of FC layers

Output layer

The output layer uses the softmax function which maps the output from the last layer into a probability for all the classes[2]. A Softmax function is a type of squashing function. Squashing functions limit the output of the function into the range 0 to 1. This allows the output to be interpreted directly as a probability.

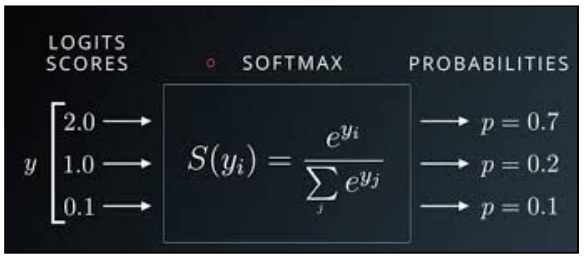

Fig 3.16 :Softmax function and the generated output probabilities

Loss Function

The output variable in a classification problem is usually a probability value f(x), called the score for the input x. Generally, the magnitude of the score represents the confidence of our prediction. The target variable y, is a binary variable, 1 for true and -1 for false.
Some classification loss algorithms are:
1. Binary Cross Entropy
2. Negative Log Likelihood
3. Margin Classifier
4. Soft Margin Classifier

This paper uses Binary Cross Entropy Loss. Cross-entropy loss increases as the predicted probability diverges from the actual label. So cross entropy makes sure we are minimizing the difference between the two probabilities.

$$L_{\text{cross-entropy}}(\hat{\mathbf{y}}, \mathbf{y}) = -\sum_i y_i \log(\hat{y}_i)$$

Fig 3.17: Formula for the calculation of Cross-entropy loss

y represents the true label distribution
$\hat{y}$ represents the predicted label distribution

Backpropagation

Back-propagation involves feeding the loss backwards in such a way that the weights in the NN can be fine tuned based on which the optimization function[3] will help the machine learn the weights that will help yield a smaller loss in the next iteration.

Fitting the model

The model is trained and tested on the respective datasets through the layers. After CNN predicts the probability of the input image belonging to a certain class, an error is calculated. The error is then back propagated and the model modifies it's weights, the feature detectors and all other trainable parameters. This is done through breaking down the training data into batches and running them through epochs[4].

```
classifier.fit_generator(
                training_set,
                samples_per_epoch = 6169 ,
                epochs = 10,
                validation_data = test_set,
                validation_steps = 1531)
```

Hyperparameter Tuning

Hyperparameters is the set of arguments whose values are set before the beginning of the learning process. They are untrainable parameters and are a part of the model selection task. Fiddling with the hyperparameter values for increasing accuracy of the model is termed as hyperparameter tuning. The tunability of an algorithm, hyperparameter, or interacting hyperparameters is a measure of how much performance can be gained by tuning it.

4. Results

This section shows the result of CNN for image classification of toddlers and pets. This paper uses classification accuracy and loss plot for easy visualization of the result. As can be seen from the classification accuracy and loss plots, our model gave fairly good validation results which are obtained by testing it on the test_set. The reason for the inconsistency in the graph lines i.e. the peak in the loss plot around the 8th epoch can be attributed to the noise in the dataset of toddler images as it was manually created. A fair attempt was made to refine the dataset and get rid of the noise.

The model gives a high classification accuracy. Validation accuracy of 93.47% was obtained and the validation loss was minimized to 0.1913.

```
96/96 [==============================] - 1553s 16s/step - loss: 0.5727 - acc: 0.7079 - val_loss: 0.4468 - val_acc: 0.7851
Epoch 2/10
96/96 [==============================] - 1556s 16s/step - loss: 0.3312 - acc: 0.8592 - val_loss: 0.2773 - val_acc: 0.8923
Epoch 3/10
96/96 [==============================] - 1577s 16s/step - loss: 0.3221 - acc: 0.8644 - val_loss: 0.2920 - val_acc: 0.8798
Epoch 4/10
96/96 [==============================] - 1549s 16s/step - loss: 0.3071 - acc: 0.8747 - val_loss: 0.2456 - val_acc: 0.9059
Epoch 5/10
96/96 [==============================] - 1574s 16s/step - loss: 0.2664 - acc: 0.8889 - val_loss: 0.2314 - val_acc: 0.9144
Epoch 6/10
96/96 [==============================] - 1565s 16s/step - loss: 0.2515 - acc: 0.9028 - val_loss: 0.2122 - val_acc: 0.9190
Epoch 7/10
96/96 [==============================] - 1557s 16s/step - loss: 0.2258 - acc: 0.9102 - val_loss: 0.2154 - val_acc: 0.9211
Epoch 8/10
96/96 [==============================] - 1557s 16s/step - loss: 0.2240 - acc: 0.9106 - val_loss: 0.3490 - val_acc: 0.8602
Epoch 9/10
96/96 [==============================] - 1572s 16s/step - loss: 0.2307 - acc: 0.9103 - val_loss: 0.4187 - val_acc: 0.8425
Epoch 10/10
96/96 [==============================] - 1555s 16s/step - loss: 0.1984 - acc: 0.9236 - val_loss: 0.1913 - val_acc: 0.9347
```

Fig 3.18:Result of training and testing of the model

Fig 3.19:Classification accuracy plot and cross-entropy loss plot

5. Discussion

The results clearly show that our model has obtained an exceptional validation accuracy which indicates that our model can be deemed fit for use in security applications. Our vision is to integrate our machine learning model with security systems to create an AI solution. This helps us increase automation and enhance security. This model can further be strengthened to incorporate multi class classification making it an asset is security solutions at various locations of mass interest.

6. Conclusion

The model was successful in learning from the given image dataset. It can successfully distinguish between human babies and pet animals such as cats and dogs. With the test set accuracy of 93.47 % we can say that our model is reliable and has not been a prey of overfitting. This can be very helpful in further making a software/application for security systems.

7. Appendix

[1] : https://www.kaggle.com/c/dogs-vs-cats

[2] : If the object belongs to multiple classes, the softmax function will not work.

[3] : This paper uses “adam” as the optimization function. More information can be found on https://towardsdatascience.com/adam-latest-trends-in-deep-learning-optimization-6be9a291375c

[4] : More information about batches and epochs can be found on Ref. no. 19.

8. References

1. https://in.mathworks.com/discovery/unsupervised-learning.html
2. https://in.mathworks.com/discovery/deep-learning.html
3. https://in.mathworks.com/discovery/neural-network.html
4. https://towardsdatascience.com/machine-learning-an-introduction-23b84d51e6d0
5. https://machinelearningmastery.com/what-is-machine-learning/
6. https://expertsystem.com/machine-learning-definition/
7. https://ml-cheatsheet.readthedocs.io/en/latest/nn_concepts.html
8. https://www.geeksforgeeks.org/effect-of-bias-in-neural-network/
9. https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html#loss-cross-entropy
10. http://www.chioka.in/differences-between-l1-and-l2-as-loss-function-and-regularization/
11. https://www.analyticsvidhya.com/blog/2020/02/cnn-vs-rnn-vs-mlp-analyzing-3-types-of-neural-networks-in-deep-learning/
12. https://towardsdatascience.com/understanding-neural-networks-from-neuron-to-rnn-cnn-and-deep-learning-cd88e90e0a90

13. https://machinelearningmastery.com/convolutional-layers-for-deep-learning-neural-networks/
14. https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks/
15. https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks-Part-2/
16. https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks
17. https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/
18. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53
19. https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/
20. https://towardsdatascience.com/activation-functions-and-its-types-which-is-better-a9a5310cc8f
21. https://towardsdatascience.com/convolutional-neural-network-17fb77e76c05
22. https://deepai.org/machine-learning-glossary-and-terms/softmax-layer
23. http://neuralnetworksanddeeplearning.com/chap2.html
24. https://towardsdatascience.com/how-does-back-propagation-in-artificial-neural-networks-work-c7cad873ea7
25. https://medium.com/@tifa2up/image-classification-using-deep-neural-networks-a-beginner-friendly-approach-using-tensorflow-94b0a090ccd4
26. https://towardsdatascience.com/wtf-is-image-classification-8e78a8235acb
27. https://www.researchgate.net/post/Combining_or_comparing_CNN_with_other_classifier
28. https://keras.io/models/model/
29. https://medium.com/data-science-group-iitr/loss-functions-and-optimization-algorithms-demystified-bb92daff331c
30. https://link.springer.com/chapter/10.1007/978-3-7908-1902-1_65
31. https://machinelearningmastery.com/keras-functional-api-deep-learning/
32. https://en.wikipedia.org/wiki/Hyperparameter_(machine_learning)
33. saama.com/different-kinds-convolutional-filters/
34. https://towardsdatascience.com/types-of-convolution-kernels-simplified-f040cb307c37
35. https://medium.com/data-science-bootcamp/understand-the-softmax-function-in-minutes-f3a59641e86d
36. Dr. Kevin Koidl, in “Loss Functions in Classification Tasks” available at https://www.scss.tcd.ie/~koidlk/cs4062/Loss-Functions.pdf
37. Deepika Jaswal, Sowmya.V and K.P.Soman, in “Image Classification Using Convolutional NeuralNetworks” available at https://www.researchgate.net/publication/275257620_Image_Classification_Using_Convolutional_Neural_Networks