**AI PROJECT DESCRIPTION**

**At first see the tutorial and read the blog**

[**https://keras.io**](https://keras.io)

[**https://www.tensorflow.org/**](https://www.tensorflow.org/)

**<https://www.youtube.com/watch?v=umGJ30-15_A>**

**etails about differnent layer---->**

[**https://keras.io/layers/convolutional/**](https://keras.io/layers/convolutional/)

**over and inderfitting**

[**https://medium.com/greyatom/what-is-underfitting-and-overfitting-in-machine-learning-and-how-to-deal-with-it-6803a989c76**](https://medium.com/greyatom/what-is-underfitting-and-overfitting-in-machine-learning-and-how-to-deal-with-it-6803a989c76)

[**https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148**](https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148)

[**https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html**](https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html)

1. **import os**

The OS module in Python provides a way of using operating system dependent

functionality.

The functions that the OS module provides allows you to interface with the

underlying operating system that Python is running on – be that Windows, Mac or

Linux.

You can find important information about your location or about the process.

In this post I will show some of these functions.

<https://www.pythonforbeginners.com/os/pythons-os-module>

***The OS module in python provides functions for interacting with the operating system.*** OS, comes under Python’s standard utility modules. This module provides a portable way of using operating system dependent functionality. The \*os\* and \*os.path\* modules include many functions to interact with the file system.

<https://www.geeksforgeeks.org/os-module-python-examples/>

1. **import matplotlib.pyplot as plt**
2. **%matplotlib inline**

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and [IPython](http://ipython.org) shells, the [Jupyter](http://jupyter.org) notebook, web application servers, and four graphical user interface toolkits.

<https://matplotlib.org/>

1. **import numpy as np**

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays- Wikipedia

NumPy is the fundamental package for scientific computing with Python. It contains among other things:

* a powerful N-dimensional array object
* sophisticated (broadcasting) functions
* tools for integrating C/C++ and Fortran code
* useful linear algebra, Fourier transform, and random number capabilities

[**http://www.numpy.org/**](http://www.numpy.org/)

1. **from sklearn.metrics import confusion\_matrix**

***sklearn.metrics. confusion\_matrix (y\_true, y\_pred, labels=None)[source] Compute confusion matrix to evaluate the accuracy of a classification***. By definition a **confusion matrix** is such that is equal to the number of observations known to be in group but predicted to be in group . Read more in the User Guide.

[**https://en.wikipedia.org/wiki/Scikit-learn**](https://en.wikipedia.org/wiki/Scikit-learn)

[**https://scikit-learn.org/stable/**](https://scikit-learn.org/stable/)

1. **from \_\_future\_\_ import print\_function**

**future**

future is the missing compatibility layer between Python 2 and Python 3. It allows you to use a single, clean Python 3.x-compatible codebase to support both Python 2 and Python 3 with minimal overhead.

[**https://pypi.org/project/future/**](https://pypi.org/project/future/)

**e.g.:**

from \_\_future\_\_ import (absolute\_import, division,

print\_function, unicode\_literals)

from builtins import (

bytes, dict, int, list, object, range, str,

ascii, chr, hex, input, next, oct, open,

pow, round, super,

filter, map, zip)

1. **import keras**
2. **from keras import backend as K**

***Keras is an*** [***open-source***](https://en.wikipedia.org/wiki/Open-source_software)[***neural-network***](https://en.wikipedia.org/wiki/Artificial_neural_network) ***library written in*** [***Python***](https://en.wikipedia.org/wiki/Python_(programming_language))*.* It is capable of running on top of [TensorFlow](https://en.wikipedia.org/wiki/TensorFlow), [Microsoft Cognitive Toolkit](https://en.wikipedia.org/wiki/Microsoft_Cognitive_Toolkit), [Theano](https://en.wikipedia.org/wiki/Theano_(software)), or [PlaidML](https://en.wikipedia.org/wiki/PlaidML).[[1]](https://en.wikipedia.org/wiki/Keras#cite_note-1)[[2]](https://en.wikipedia.org/wiki/Keras#cite_note-2) Designed to enable fast experimentation with [deep neural networks](https://en.wikipedia.org/wiki/Deep_learning), it focuses on being user-friendly, modular, and extensible. It was developed as part of the research effort of project ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System),[[3]](https://en.wikipedia.org/wiki/Keras#cite_note-3) and its primary author and maintainer is François Chollet, a [Google](https://en.wikipedia.org/wiki/Google) engineer. Chollet also is the author of the XCeption deep neural network model[[4]](https://en.wikipedia.org/wiki/Keras#cite_note-4).

[**https://en.wikipedia.org/wiki/Keras**](https://en.wikipedia.org/wiki/Keras)

***Keras is a high-level neural networks API, written in Python and capable of running on top of*** [***TensorFlow***](https://github.com/tensorflow/tensorflow)***,*** [***CNTK***](https://github.com/Microsoft/cntk)***, or*** [***Theano***](https://github.com/Theano/Theano). It was developed with a focus on enabling fast experimentation. Being able to go from idea to result with the least possible delay is key to doing good research.

Use Keras if you need a deep learning library that:

* Allows for easy and fast prototyping (through user friendliness, modularity, and extensibility).
* Supports both convolutional networks and recurrent networks, as well as combinations of the two.
* Runs seamlessly on CPU and GPU.

[**https://keras.io/**](https://keras.io/)

1. **from keras import backend as K**

Keras is a model-level library, providing high-level building blocks for developing deep learning models. ***It does not handle low-level operations such as tensor products, convolutions and so on itself. Instead, it relies on a specialized, well optimized tensor manipulation library to do so, serving as the "backend engine" of Keras***. Rather than picking one single tensor library and making the implementation of Keras tied to that library, Keras handles the problem in a modular way, and several different backend engines can be plugged seamlessly into Keras.

At this time, Keras has three backend implementations available: the **TensorFlow** backend, the **Theano** backend, and the **CNTK** backend.

* [***TensorFlow***](http://www.tensorflow.org/) ***is an open-source symbolic tensor manipulation framework developed by Google.***
* [Theano](http://deeplearning.net/software/theano/) is an open-source symbolic tensor manipulation framework developed by LISA Lab at Université de Montréal.
* [CNTK](https://www.microsoft.com/en-us/cognitive-toolkit/) is an open-source toolkit for deep learning developed by Microsoft.

In the future, we are likely to add more backend options.

[**https://keras.io/backend/**](https://keras.io/backend/)

1. **from IPython.display import SVG**

[**https://ipython.org/ipython-doc/3/api/generated/IPython.display.html**](https://ipython.org/ipython-doc/3/api/generated/IPython.display.html)

Scalable Vector Graphics(SVG)

Scalable Vector Graphics is an XML-based vector image format for two-dimensional graphics with support for interactivity and animation. The SVG specification is an open standard developed by the World Wide Web Consortium since 1999. SVG images and their behaviors are defined in XML text files. [Wikipedia](https://en.wikipedia.org/wiki/Scalable_Vector_Graphics)

1. **from keras.optimizers import Adam**

Adam is an optimization algorithm that can used instead of the classical stochastic gradient descent procedure to update network weights iterative based in training data.

Adam was presented by [Diederik Kingma](http://dpkingma.com/) from OpenAI and [Jimmy Ba](https://jimmylba.github.io/) from the University of Toronto in their 2015 [ICLR](http://www.iclr.cc/doku.php?id=iclr2015:main) paper (poster) titled “[Adam: A Method for Stochastic Optimization](https://arxiv.org/abs/1412.6980)“. I will quote liberally from their paper in this post, unless stated otherwise.

The algorithm is called Adam. It is not an acronym and is not written as “ADAM”.

… ***the name Adam is derived from adaptive moment estimation.***

When introducing the algorithm, the authors list the attractive benefits of using Adam on non-convex optimization problems, as follows:

* Straightforward to implement.
* Computationally efficient.
* Little memory requirements.
* Invariant to diagonal rescale of the gradients.
* Well suited for problems that are large in terms of data and/or parameters.
* Appropriate for non-stationary objectives.
* Appropriate for problems with very noisy/or sparse gradients.

[**https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/**](https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/)

[**https://www.tensorflow.org/api\_docs/python/tf/keras/optimizers/Adam**](https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/Adam)

[**https://keras.io/optimizers/**](https://keras.io/optimizers/)

1. **from keras.utils import plot\_model**

The *plot\_model()* function in Keras will create a plot of your network. This function takes a few useful arguments:

* *model*: (required) The model that you wish to plot.
* *to\_file*: (required) The name of the file to which to save the plot.
* *show\_shapes*: (optional, defaults to *False*) Whether or not to show the output shapes of each layer.
* *show\_layer\_names*: (optional, defaults to *True*) Whether or not to show the name for each layer.

**https://machinelearningmastery.com/visualize-deep-learning-neural-network-model-keras/**

[**https://keras.io/visualization/**](https://keras.io/visualization/)

[**https://www.tensorflow.org/api\_docs/python/tf/keras/utils/plot\_model**](https://www.tensorflow.org/api_docs/python/tf/keras/utils/plot_model)

1. **from keras.models import Model, Sequential**

[**https://keras.io/models/model/**](https://keras.io/models/model/)

1. **from keras.callbacks import ReduceLROnPlateau**

## Class ReduceLROnPlateau

Inherits From: [Callback](https://www.tensorflow.org/api_docs/python/tf/keras/callbacks/Callback)

Defined in [tensorflow/python/keras/callbacks.py](https://www.tensorflow.org/code/stable/tensorflow/python/keras/callbacks.py).

Reduce learning rate when a metric has stopped improving.

Models often benefit from reducing the learning rate by a factor of 2-10 once learning stagnates. This callback monitors a quantity and if no improvement is seen for a 'patience' number of epochs, the learning rate is reduced.

[**https://www.tensorflow.org/api\_docs/python/tf/keras/callbacks/ReduceLROnPlateau**](https://www.tensorflow.org/api_docs/python/tf/keras/callbacks/ReduceLROnPlateau)

[**https://keras.io/callbacks/**](https://keras.io/callbacks/)

**# Agrument:**

* **monitor**: quantity to be monitored.
* **factor**: factor by which the learning rate will be reduced. new\_lr = lr \* factor
* **patience**: number of epochs with no improvement after which learning rate will be reduced.
* **verbose**: int. 0: quiet, 1: update messages.
* **mode**: one of {auto, min, max}. In min mode, lr will be reduced when the quantity monitored has stopped decreasing; in max mode it will be reduced when the quantity monitored has stopped increasing; in auto mode, the direction is automatically inferred from the name of the monitored quantity.
* **min\_delta**: threshold for measuring the new optimum, to only focus on significant changes.
* **cooldown**: number of epochs to wait before resuming normal operation after lr has been reduced.
* **min\_lr**: lower bound on the learning rate.

1. **from keras.utils.vis\_utils import model\_to\_dot**

**Model visualization**

[**https://keras.io/visualization/**](https://keras.io/visualization/)

1. **from keras.utils.generic\_utils import get\_custom\_objects**

# tf.keras.utils.get\_custom\_objects

tf.keras.utils.get\_custom\_objects()

Defined in [tensorflow/python/keras/utils/generic\_utils.py](https://www.tensorflow.org/code/stable/tensorflow/python/keras/utils/generic_utils.py).

Retrieves a live reference to the global dictionary of custom objects.

Updating and clearing custom objects using custom\_object\_scope is preferred, but get\_custom\_objects can be used to directly access \_GLOBAL\_CUSTOM\_OBJECTS.

Example:

    get\_custom\_objects().clear()  
    get\_custom\_objects()['MyObject'] = MyObject

1. **from keras.preprocessing.image import ImageDataGenerator**

[**https://medium.com/@arindambaidya168/https-medium-com-arindambaidya168-using-keras-imagedatagenerator-b94a87cdefad**](https://medium.com/@arindambaidya168/https-medium-com-arindambaidya168-using-keras-imagedatagenerator-b94a87cdefad)

1. **from keras.layers import Input, Conv2D, Dense, Flatten, MaxPool2D**

[**https://keras.io/layers/convolutional/**](https://keras.io/layers/convolutional/)

[**https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148**](https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148)

1. **from keras.layers import Activation, Add, BatchNormalization, Dropout**

**activation = relu**

[**https://keras.io/layers/core/**](https://keras.io/layers/core/)

[**https://keras.io/layers/normalization/**](https://keras.io/layers/normalization/)

1. **model = Sequential()**
3. **model.add(Conv2D(activation ='relu', input\_shape = (64,64,3), filters=64, kernel\_size=(3, 3), padding="SAME", strides=(1, 1)))**
5. **#model.add(Conv2D(activation ='relu',filters=32, kernel\_size=(3, 3), padding="SAME", strides=(1,1)))**
6. **#model.add(MaxPool2D(pool\_size=(2, 2), strides=(2, 2)))**
8. **model.add(Conv2D(activation ='relu', filters=64, kernel\_size=(3, 3), padding="SAME", strides=(1, 1)))**
9. **model.add(MaxPool2D(pool\_size=(2, 2), strides=(2, 2)))**
10. **model.add(Conv2D(activation ='relu',filters=128, kernel\_size=(3, 3), padding="SAME", strides=(1,1)))**
11. **model.add(Conv2D(activation ='relu',filters=128, kernel\_size=(3, 3), padding="SAME", strides=(1,1)))**
12. **model.add(MaxPool2D(pool\_size=(2, 2), strides=(2, 2)))**
14. **# model.add(Dropout(0.25))**
15. **model.add(BatchNormalization())**
17. **model.add(Conv2D(activation ='relu',filters=256, kernel\_size=(3, 3), padding="SAME", strides=(1,1)))**
18. **model.add(Conv2D(activation ='relu',filters=256, kernel\_size=(3, 3), padding="SAME", strides=(1,1)))**
19. **model.add(MaxPool2D(pool\_size=(2, 2), strides=(2, 2)))**
20. **#model.add(Conv2D(activation ='relu',filters=512, kernel\_size=(3, 3), padding="SAME", strides=(1,1)))**
21. **#model.add(Conv2D(activation ='relu',filters=512, kernel\_size=(3, 3), padding="SAME", strides=(1,1)))**
22. **#model.add(MaxPool2D(pool\_size=(2, 2), strides=(2, 2)))**
24. **model.add(Flatten())**
25. **model.add(Dense(512, activation='relu'))**
26. **model.add(Dropout(0.5))**
27. **model.add(Dense(units = 10, activation='softmax'))**
28. **#show model summary**
29. **model.summary()**

[**https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148**](https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148)

[**https://www.youtube.com/watch?v=umGJ30-15\_A**](https://www.youtube.com/watch?v=umGJ30-15_A)

There are two ways to build Keras models: *sequential* and *functional*.

The **sequential** API allows you to create models layer-by-layer for most problems. It is limited in that it does not allow you to create models that share layers or have multiple inputs or outputs.

Alternatively, the **functional** API allows you to create models that have a lot more flexibility as you can easily define models where layers connect to more than just the previous and next layers. In fact, you can connect layers to (literally) any other layer. As a result, creating complex networks such as siamese networks and residual networks become possible.

[**https://jovianlin.io/keras-models-sequential-vs-functional/**](https://jovianlin.io/keras-models-sequential-vs-functional/)

1. **optimizer = Adam(lr=0.001)**
2. **model.compile(loss='categorical\_crossentropy', optimizer = optimizer, metrics=["accuracy"])**
3. **train\_datagen = ImageDataGenerator(**

**rotation\_range=40,**

**width\_shift\_range=0.2,**

**height\_shift\_range=0.2,**

**rescale=1./255,**

**shear\_range=0.2,**

**zoom\_range=0.2,**

**horizontal\_flip=True,**

**fill\_mode='nearest')**

**test\_datagen = ImageDataGenerator(rotation\_range=40,**

**width\_shift\_range=0.2,**

**height\_shift\_range=0.2,**

**rescale=1./255,**

**shear\_range=0.2,**

**zoom\_range=0.2,**

**horizontal\_flip=True,**

**fill\_mode='nearest')**

[**https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html**](https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html)

**#SEE in th number 11**

**train\_generator = train\_datagen.flow\_from\_directory(**

**'plantRecognitionByLeaf-master/dataset/dataset\_original/train\_data',**

**target\_size=(64, 64),**

**batch\_size=30,**

**color\_mode='rgb',**

**class\_mode='categorical')**

**validation\_generator = test\_datagen.flow\_from\_directory(**

**'plantRecognitionByLeaf-master/dataset/dataset\_original/test\_data',**

**target\_size=(64, 64),**

**batch\_size=30,**

**color\_mode='rgb',**

**class\_mode='categorical')**

[**https://medium.com/@vijayabhaskar96/tutorial-image-classification-with-keras-flow-from-directory-and-generators-95f75ebe5720**](https://medium.com/@vijayabhaskar96/tutorial-image-classification-with-keras-flow-from-directory-and-generators-95f75ebe5720)

## **code: learning\_rate\_reduction = ReduceLROnPlateau(monitor='val\_acc',**

## **patience=3,**

## **verbose=1,**

## **factor=0.5,**

## **min\_lr=0.000001)**

## Class ReduceLROnPlateau

Inherits From: [Callback](https://www.tensorflow.org/api_docs/python/tf/keras/callbacks/Callback)

Defined in [tensorflow/python/keras/callbacks.py](https://www.tensorflow.org/code/stable/tensorflow/python/keras/callbacks.py).

Reduce learning rate when a metric has stopped improving.

Models often benefit from reducing the learning rate by a factor of 2-10 once learning stagnates. This callback monitors a quantity and if no improvement is seen for a 'patience' number of epochs, the learning rate is reduced.

[**https://www.tensorflow.org/api\_docs/python/tf/keras/callbacks/ReduceLROnPlateau**](https://www.tensorflow.org/api_docs/python/tf/keras/callbacks/ReduceLROnPlateau)

[**https://keras.io/callbacks/**](https://keras.io/callbacks/)

**# Agrument:**

* **monitor**: quantity to be monitored.
* **factor**: factor by which the learning rate will be reduced. new\_lr = lr \* factor
* **patience**: number of epochs with no improvement after which learning rate will be reduced.
* **verbose**: int. 0: quiet, 1: update messages.
* **mode**: one of {auto, min, max}. In min mode, lr will be reduced when the quantity monitored has stopped decreasing; in max mode it will be reduced when the quantity monitored has stopped increasing; in auto mode, the direction is automatically inferred from the name of the monitored quantity.
* **min\_delta**: threshold for measuring the new optimum, to only focus on significant changes.
* **cooldown**: number of epochs to wait before resuming normal operation after lr has been reduced.
* **min\_lr**: lower bound on the learning rate.