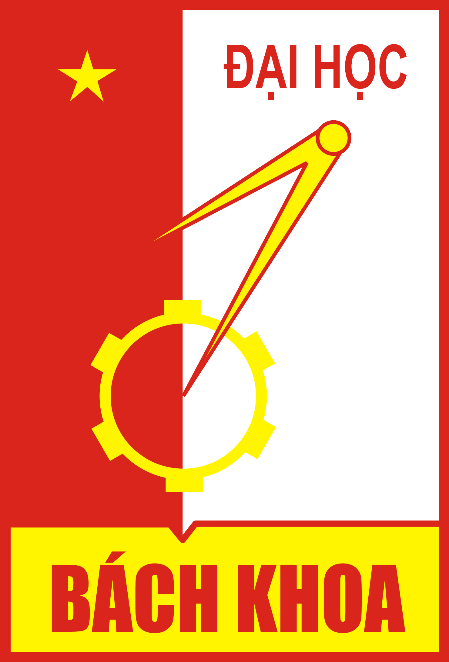
HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

SCHOOL OF INFORMATION AND TECHNOLOGY

-----🙞🙜🕮🙞🙜-----

****

INTRODUCTION TO DATA SCIENCE REPORT

Group 7

Class: ICT 01 K61

Phạm Hồng Quang – 20163316

Nguyễn Văn Nam – 20162844

Trần Tùng Dương – 20165933

Ngô Minh Quang - 20163303

Nguyễn Đức Nhật - 20163040

*Under supervision of*

*Dr. Thân Quang Khoát, Ph.D*

*Hanoi, 2019*

MIDTERM PROJECT REPORT

USING CONVOLUTIONAL NEURAL NETWORKS FOR IMAGE RECOGNITION

**I. Problems:**

Convolutional neural network (CNN) is extremely popular and appears to be the best in image-classifying problems. Being able to predict what an image is, for example, recognizing hand-written digits is what brings computers to another level, paving the way for innovations and unprecedented breakthroughs in the future. Realizing the importance of training computers to recognize and classify objects, as these areas are delivering the current state-of-the-art in recognition tasks like speech recognition, we choose our mid-term project to specify in deep network for image recognition using CNNs.

**II. Data set and libraries used**

The data set to be used in the project is CIFAR-10, which is widely used in machine learning research. We will base on CIFAR-10 to build and train a deep CNN model with multiple layers using the libraries:

- Keras: Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow allowing for easy and fast prototyping. We use keras to pre-process images and support training and testing models.

- Matplotlib: We use this library to fast drawing graphs and plot images

- Pandas: To provide high-performance, easy-to-use data structures and data analysis tools.

- Scipy: To process images

- Sklearn: Evaluation of models that are trained and created

- Cv2: Read and import images

- Numpy: To manipulate arrays for processing between layers

**About CIFAR-10 dataset**

The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.

The dataset is divided into five training batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly-selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class.  
  
Here are the classes in the dataset, as well as 10 random images from each:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| airplane | https://www.cs.toronto.edu/~kriz/cifar-10-sample/airplane1.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/airplane2.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/airplane3.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/airplane4.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/airplane5.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/airplane6.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/airplane7.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/airplane8.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/airplane9.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/airplane10.png |
| automobile | https://www.cs.toronto.edu/~kriz/cifar-10-sample/automobile1.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/automobile2.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/automobile3.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/automobile4.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/automobile5.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/automobile6.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/automobile7.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/automobile8.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/automobile9.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/automobile10.png |
| bird | https://www.cs.toronto.edu/~kriz/cifar-10-sample/bird1.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/bird2.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/bird3.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/bird4.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/bird5.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/bird6.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/bird7.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/bird8.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/bird9.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/bird10.png |
| cat | https://www.cs.toronto.edu/~kriz/cifar-10-sample/cat1.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/cat2.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/cat3.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/cat4.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/cat5.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/cat6.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/cat7.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/cat8.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/cat9.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/cat10.png |
| deer | https://www.cs.toronto.edu/~kriz/cifar-10-sample/deer1.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/deer2.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/deer3.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/deer4.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/deer5.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/deer6.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/deer7.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/deer8.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/deer9.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/deer10.png |
| dog | https://www.cs.toronto.edu/~kriz/cifar-10-sample/dog1.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/dog2.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/dog3.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/dog4.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/dog5.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/dog6.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/dog7.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/dog8.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/dog9.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/dog10.png |
| frog | https://www.cs.toronto.edu/~kriz/cifar-10-sample/frog1.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/frog2.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/frog3.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/frog4.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/frog5.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/frog6.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/frog7.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/frog8.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/frog9.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/frog10.png |
| horse | https://www.cs.toronto.edu/~kriz/cifar-10-sample/horse1.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/horse2.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/horse3.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/horse4.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/horse5.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/horse6.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/horse7.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/horse8.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/horse9.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/horse10.png |
| ship | https://www.cs.toronto.edu/~kriz/cifar-10-sample/ship1.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/ship2.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/ship3.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/ship4.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/ship5.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/ship6.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/ship7.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/ship8.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/ship9.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/ship10.png |
| truck | https://www.cs.toronto.edu/~kriz/cifar-10-sample/truck1.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/truck2.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/truck3.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/truck4.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/truck5.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/truck6.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/truck7.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/truck8.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/truck9.png | https://www.cs.toronto.edu/~kriz/cifar-10-sample/truck10.png |
|  |  |  |  |  |  |  |  |  |  |  |

The classes are completely mutually exclusive. There is no overlap between automobiles and trucks. "Automobile" includes sedans, SUVs, things of that sort. "Truck" includes only big trucks. Neither includes pickup trucks.

**III. Solutions:**

**Import and pre-processing images**

Images must be Z-score (mean-standard deviation) normalized because in the process of training our network, we're going to be multiplying weights and adding to biases these initial inputs in order to cause activations that we then backpropogate with the gradients to train the model. Z-score normalization is important because it results in similarly-ranged feature values and that the gradients don’t go out of control (need one global learning rate multiplier).

CIFAR-10 dataset represents default target data type as integer encoding: Airplane - 0, automobile - 1, bird - 2, cat - 3, deer - 4, dog - 5, frog - 6, horse - 7, ship - 8, truck - 9

We therefore would want to convert target vector to one-hot coding as form of binary matrices since there is no relationship between each of the 10 classes in the CIFA-10 dataset:

For example: An array of data type as follow

array([0, 2, 1, 2, 0]). We use keras.utils.to\_categorical to produce the following matrix

array([[ 1., 0., 0.],

[ 0., 0., 1.],

[ 0., 1., 0.],

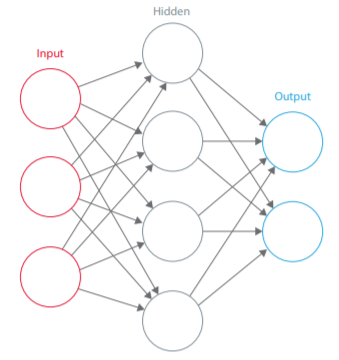
[ 0., 0., 1.],

[ 1., 0., 0.]])

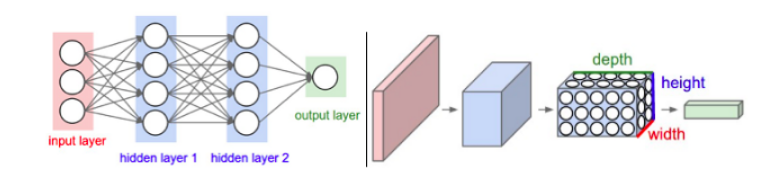
**Building a convolutional neural network**

What is a CNN?

A neural network is a system of interconnected artificial “neurons” that exchange messages between each other. The connections have numeric weights (representing the connection between each node) that are tuned during the training process, so that a properly trained network will respond correctly when presented with an image or pattern to recognize. The network consists of multiple layers of feature-detecting “neurons”. Each layer has many neurons that respond to different combinations of inputs from the previous layers. As shown in below Figure, the layers are built up so that the first layer detects a set of primitive patterns in the input, the second layer detects patterns of patterns, the third layer detects patterns of those patterns, and so on.



Convolutional Neural Networks, the layers are organized in 3 dimensions: width, height and depth. The neurons in one layer do not connect to all the neurons in the next layer but only to a small region of it. The final output will be reduced to a single vector of probability scores, organized along the depth dimension.



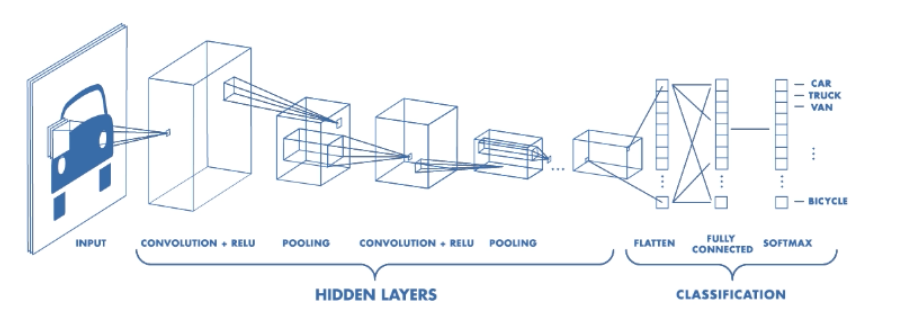
CNNs have two components:

**The Hidden layers/Feature extraction part**

In this part, the network will perform a series of convolutions and pooling operations during which the features are detected. If you had a picture of a zebra, this is the part where the network would recognize its stripes, two ears, and four legs.

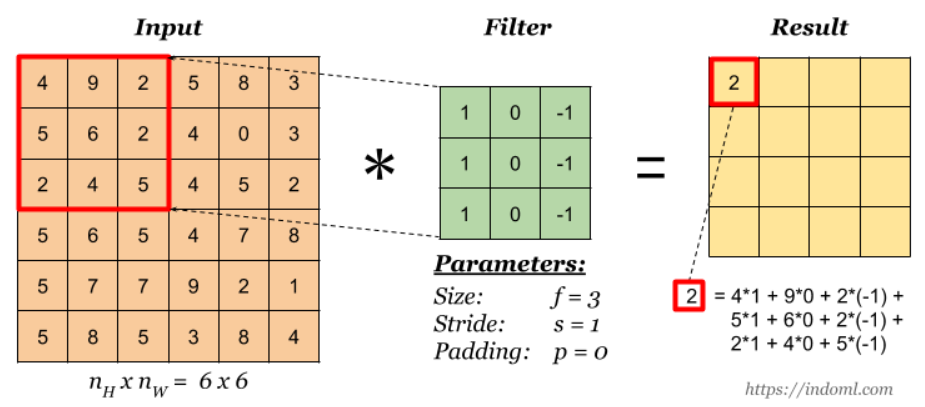
**The Classification part**

Here, the fully connected layers will serve as a classifier on top of these extracted features. They will assign a probability for the object on the image being what the algorithm predicts it is.



**Feature extraction**

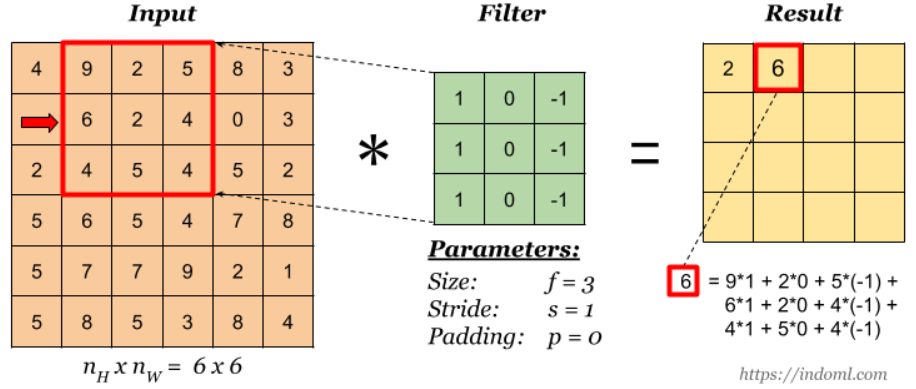
Convolution is one of the main building blocks of a CNN. The term convolution refers to the mathematical combination of two functions to produce a third function. It merges two sets of information.

In the case of a CNN, the convolution is performed on the input data with the use of a filter or kernel (these terms are used interchangeably) to then produce a feature map. We execute a convolution by sliding the filter over the input. At every location, a matrix multiplication is performed and sums the result onto the feature map.

We perfom numerous convolutions on our input, where each operation uses a different filter. This results in different feature maps. In the end, we take all of these feature maps and put them together as the final output of the convolution layer.

We use an activation function to make our output non-linear. In the case of a Convolutional Neural Network, the output of the convolution will be passed through the activation function. In the project, we use Exponential Linear Units (ELU)

Stride is the size of the step the convolution filter moves each time. A stride size is usually 1, meaning the filter slides pixel by pixel. By increasing the stride size, the filter is sliding over the input with a larger interval and thus has less overlap between the cells.



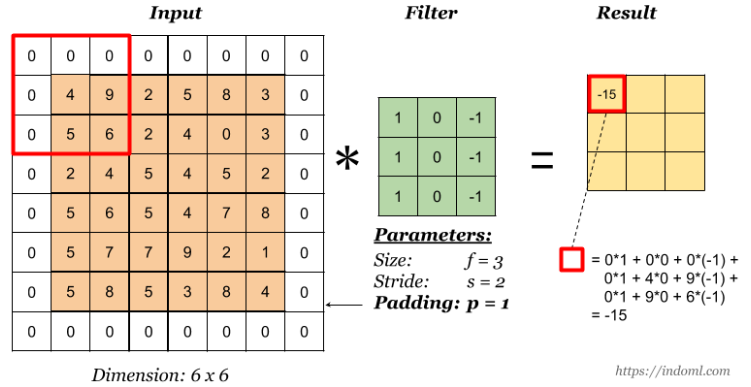
If stride is set to equal to 2

A screenshot of a cell phone

Description automatically generated

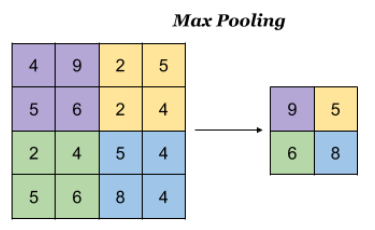
Because the size of the feature map is always smaller than the input, we have to do something to prevent our feature map from shrinking. This is where we use padding.

A layer of zero-value pixels is added to surround the input with zeros, so that our feature map will not shrink. In addition to keeping the spatial size constant after performing convolution, padding also improves performance and makes sure the kernel and stride size will fit in the input.



After a convolution layer, it is common to add a pooling layer in between CNN layers. The function of pooling is to continuously reduce the dimensionality to reduce the number of parameters and computation in the network. This shortens the training time and controls overfitting.

The most frequent type of pooling is max pooling, which takes the maximum value in each window. These window sizes need to be specified beforehand. This decreases the feature map size while at the same time keeping the significant information.



**Classification**

After the convolution and pooling layers, our classification part consists of a few fully connected layers. However, these fully connected layers can only accept 1 Dimensional data. To convert our 3D data to 1D, we use the function flatten in Python. This essentially arranges our 3D volume into a 1D vector. The last layers of a Convolutional NN are fully connected layers. Neurons in a fully connected layer have full connections to all the activations in the previous layer.

**IV. Evaluation**

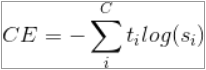
We calculate the accuracy using the confusion matrix

Below is the table of confusion after predicting 10000 images in test dataset

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Predicted** | | | | | | | | | | |
| **Target** | **Class** | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **0** | **880** | 14 | 20 | 5 | 7 | 2 | 14 | 10 | 22 | 26 |
| **1** | 6 | **947** | 0 | 0 | 0 | 1 | 4 | 0 | 6 | 36 |
| **2** | 61 | 0 | **656** | 21 | 49 | 47 | 139 | 17 | 3 | 7 |
| **3** | 20 | 6 | 17 | **619** | 47 | 145 | 100 | 27 | 5 | 14 |
| **4** | 12 | 1 | 17 | 31 | **775** | 12 | 111 | 39 | 2 | 0 |
| **5** | 9 | 0 | 19 | 76 | 32 | **777** | 37 | 38 | 1 | 11 |
| **6** | 4 | 1 | 7 | 17 | 1 | 5 | **959** | 3 | 2 | 1 |
| **7** | 16 | 0 | 9 | 15 | 25 | 27 | 19 | **882** | 0 | 7 |
| **8** | 45 | 32 | 2 | 2 | 2 | 2 | 14 | 4 | **878** | 19 |
| **9** | 15 | 55 | 2 | 4 | 0 | 0 | 2 | 4 | 6 | **912** |

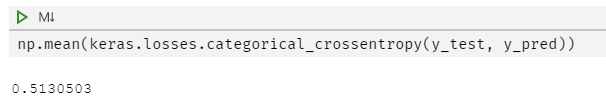
Accuracy = = 0.8285

We calculate the loss using the Cross-Entropy loss equation



With C – number of classes, – target value, – predicted value

Since we predicted 10000 images, the final CE loss value would be the average of all loss values of 10000 predictions. Using keras’ built-in function categorical\_crossentropy, final CE loss is calculated as the following:

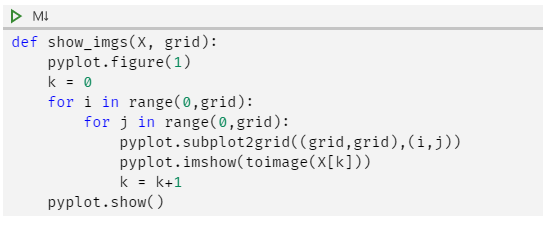


**V. Demonstration**

First, import the libraries



Plot the images





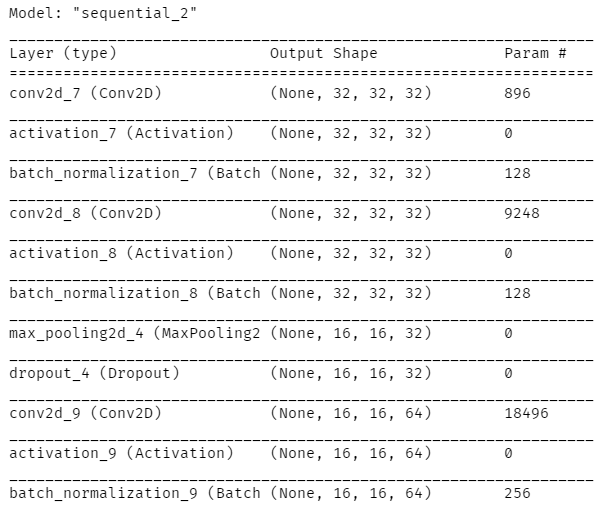
Import and pre-process data



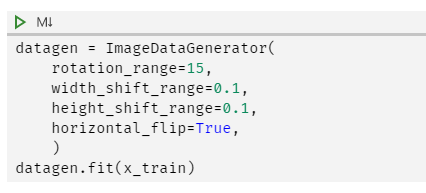
Build CNN model using Keras



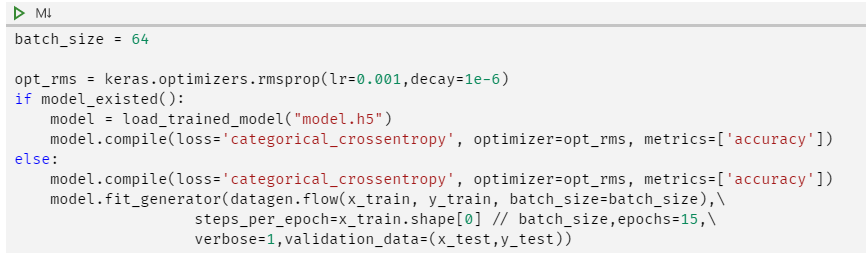
Result:



Augmenting data



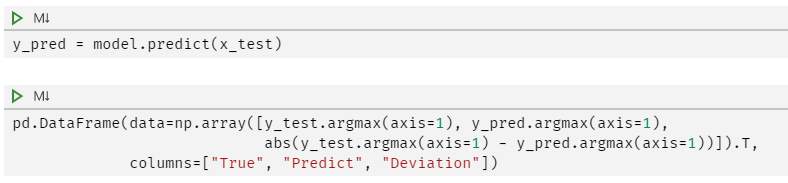
Train model or load if existed



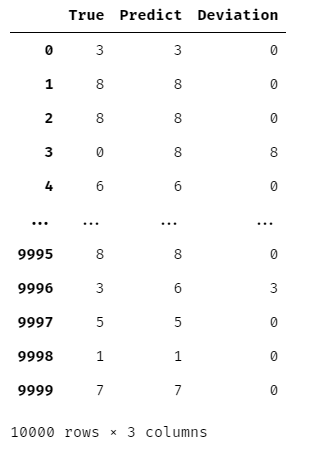
Saving model to disk



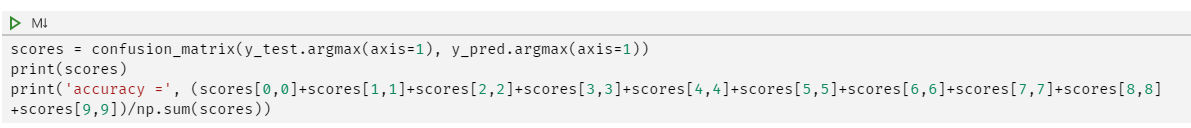
EVALUATION: Predict all test images



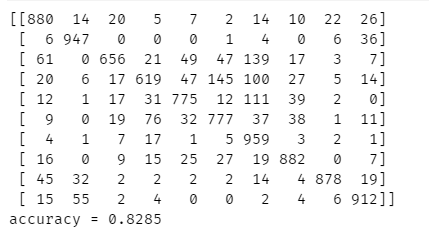
Result:



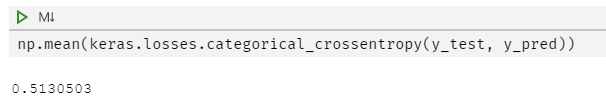
Evaluate model’s accuracy on test dataset using confusion matrix



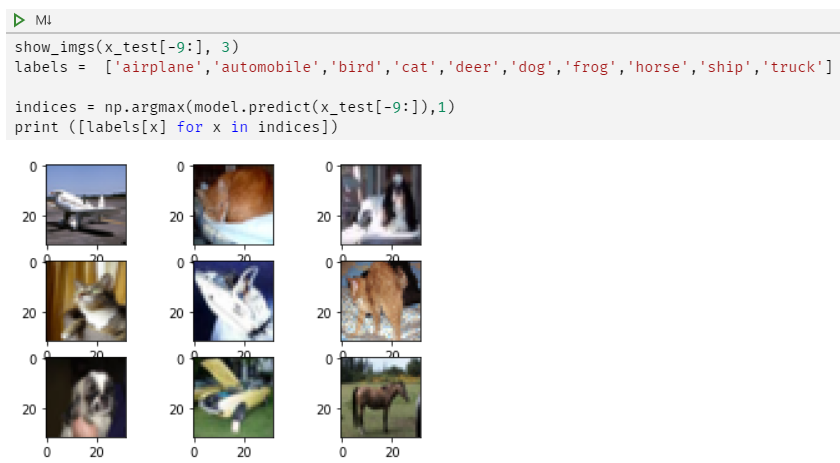
Result:



Calculate categorical cross-entropy loss



Predict the last 9 images



Result: 

Predict custom image:



Result:

