

KERAS AND APPLICATION FOR IMAGE CLASSIFICATION

Tran Thi Thu Phuong

Hanoi Metropolitan University

Abstract: *Keras is a deep learning API written in Python, it is developed with the aim of enabling fast experimentation. This paper presents a method using CNN model in keras framework to classify handwritten numbers. The model applied is simple to use and the result obtained is very precise. The result of this study provides a helpful method to solve other image classification issues.*

Keywords: *Algorithm, computer vision, CNN, image classification, Keras, model.*

Received 12.1.2021; accepted for publication 25.1.2021

Email: ttphuong2@daihocthudo.edu.vn

1. INTRODUCTION

Computer Vision (CV) is defined as a field of study that seeks to develop techniques to help computers “see” and understand the content of digital images such as photographs and videos [1]. It has many applications, including self-driving cars, traffic monitoring, and facial recognition. One of the approaches of Artificial Intelligence of the CV problems is Deep Learning that can be classified in many problems such as image classification, object detection, segmentation, and class of generating images. The essence of the Deep Learning problems is that the computer learns from existing data and then predicts new data. There are some basic steps to solve a Deep Learning problem:

- Step 1: Constructing the problem
- Step 2: Prepare data (dataset)
- Step 3: Build the model
- Step 4: Define the loss function
- Step 5: Do a backpropagation and apply a gradient descent to find the parameters including weight and bias to optimize loss function.
- Step 6: Predict the new data by model with the coefficients found above.

When building the model, applying the knowledge of neural network and convolutional neural network, we can build a complete model using python. However the back propagation step becomes very complicated. It is difficult to implement and optimize the speed calculation. This is why deep learning frameworks come into existence with some features. Users only need to define the model and the loss function, the framework will take care of the backpropagation. The definition of layers, activation function, and loss function is simpler for users.

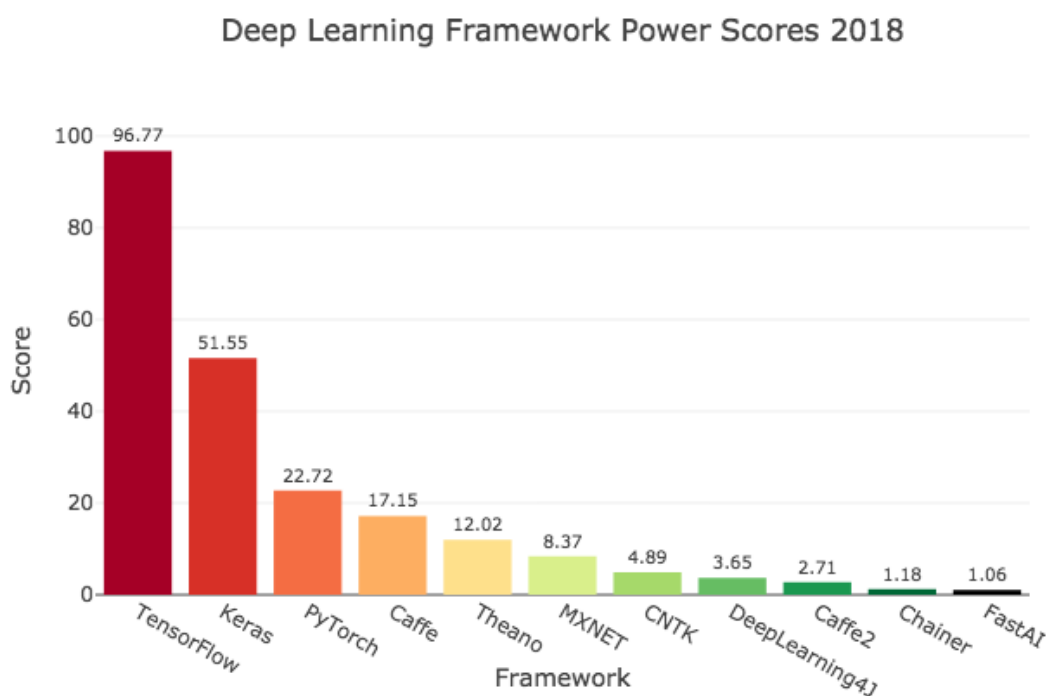


Figure 1. The popular Deep Learning frameworks [2]

The figure 1 presents some popular Deep Learning frameworks. It can be seen that tensorflow is the most popular framework, however tensorflow is quite difficult to use. So this paper proposes a method to use query framework which easy to use, user - friendly, but good enough to solve image classification problems.

2. CONTENT

2.1. Image classification and the proposed method

2.1.1. What is image classification?

In the field of Deep Learning, image classification is defined as where a computer can analyze an image and identify the ‘class’ the image falls under. (Or a probability of the image

being part of a ‘class’)[3]. A class is essentially a label, for instance, ‘car’, ‘animal’, ‘building’ and so on. For example, you input an image of a dog. Image classification is the process of the computer analyzing the image and telling you it’s a dog. (Or the probability that it’s a dog.)

There are some machine learning image classification algorithms. For example, K-Nearest Neighbors that is a classification algorithm that examines the closest training examples and looks at their labels to ascertain the most probable label for a given test example. The algorithm is extremely simple and it deals with multiple classes quite easily. However, the similarity calculation based on all features equally can be prone to mis_classification when provided a subset of the important features for the image classification.

Support Vector Machines (SVM) are a classification method that places points in space and then draws dividing lines between the points, placing objects in different classes depending on which side of the dividing plane the points fall on. A drawback of SVM classifiers is that they tend to be limited by both size and speed, with speed suffering as size increases.

The most popularly used image classification algorithms now is the Convolutional Neural Network (CNN). CNN are customized versions of neural networks that combine the multilayer neural networks with specialized layers that are capable of extracting the most important features and relevant to the classification of an object.

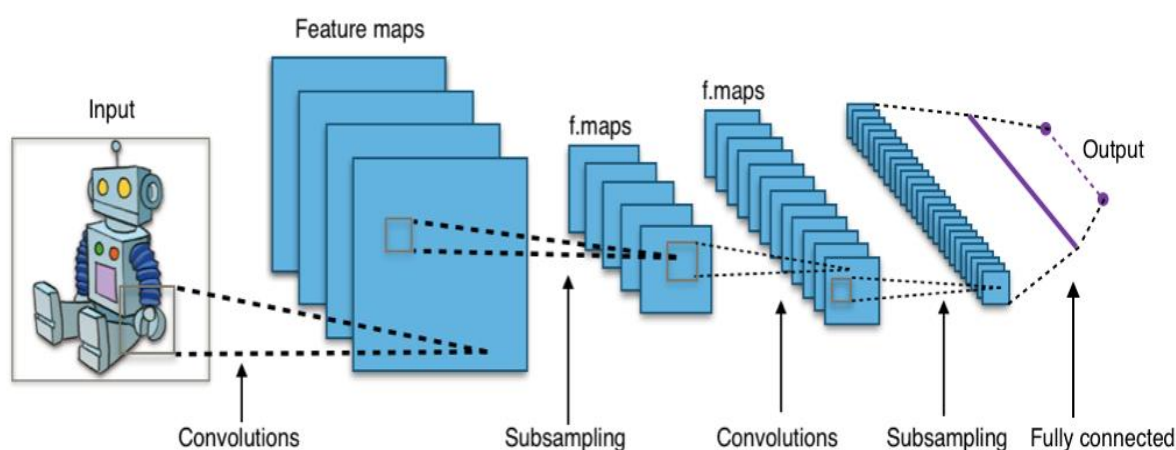


Figure 2. Full convolutional neural network via Wikimedia Commons

Convolutional Neural Networks are comprised of some different parts. The convolutional layers are what extracting the features of the image and convert them into a format that the neural network layers can interpret and learn from. The early convolutional

layers are responsible for extracting the most basic elements of the image, like simple lines and boundaries. The middle convolutional layers begin to capture more complex shapes, like simple curves and corners. The later, deeper convolutional layers extract the high-level features of the image.

The advantages of CNN are automatically discovering, generating, and learning features of the images. This greatly reduces the need to manually label and segment images to prepare them for machine learning algorithms. Therefore, the section below presents the method use CNN algorithm for image classification with keras framework.

2.1.2. Image classification with keras framework

2.1.2.1 Constructing the problem

Suppose that we have a $28 * 28$ gray image of numbers 0 to 9. After using CNN algorithm with keras framework, the program must answer what number it is.

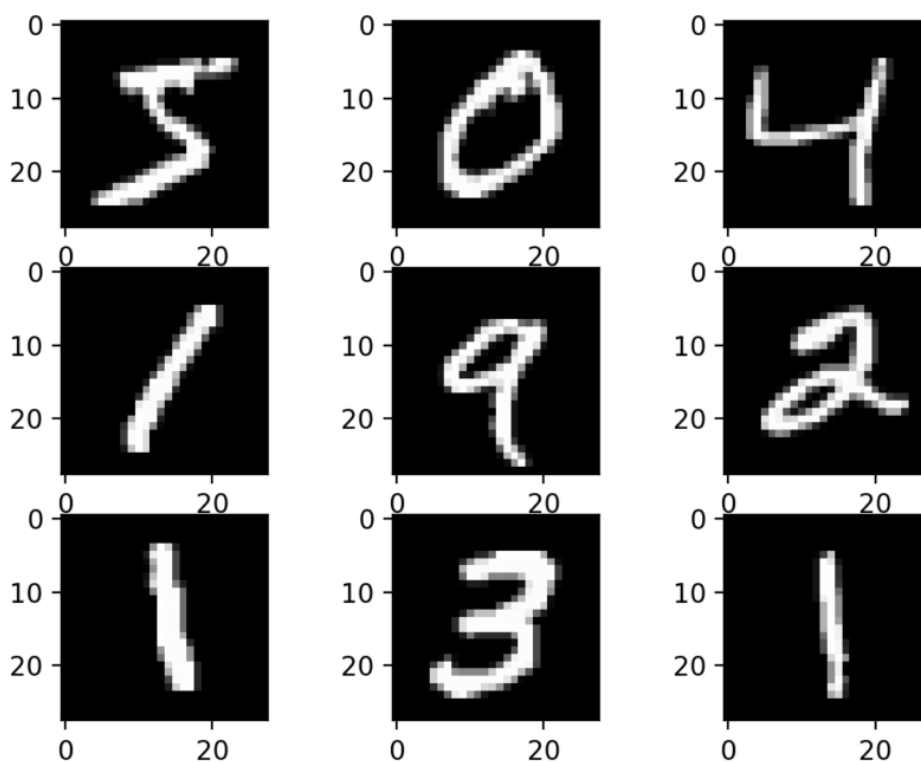


Figure 3. The example of handwritten digit number to predict

2.1.2.2 Prepare data (dataset)

The goal of machine learning is not to predict well with the available data, it has to make good predictions of the new data in the real world. The dataset used in this case is MINIST which includes 2 subsets: training set includes 60,000 photos, handwritten numerals and test

set of 10,000 image numbers. Then in the training set, we will divide 50,000 data for the training set and 10,000 data for the validation set, retain 10,000 test set data.

2.1.2.3 Build the model

The general of CNN algorithm is that Input image \rightarrow Convolutional layer (Conv) + Pooling layer (Pool) \rightarrow Fully connected layer (FC) \rightarrow Output. Similar to logistic regression, instead of showing what number is, we predict the percentage of the number image, for example: 90% of the image is number 5, 1% of the photo is number 1,...

Based on neural network, each layer will perform 2 steps: total linear nodes in the previous layer and performs the activation function (sigmoid function, softmax function). Due after the summation step linear gives real values so need to use softmax function used to convert the real value in the nodes on the output layer to a percentage value.. Since each image will belong to a class from 0 to 9, there will be 10 classes in total. So the output layer will have 10 nodes to correspond to the image percentage is 0.1,..., 9.

2.1.2.4 Define the loss function

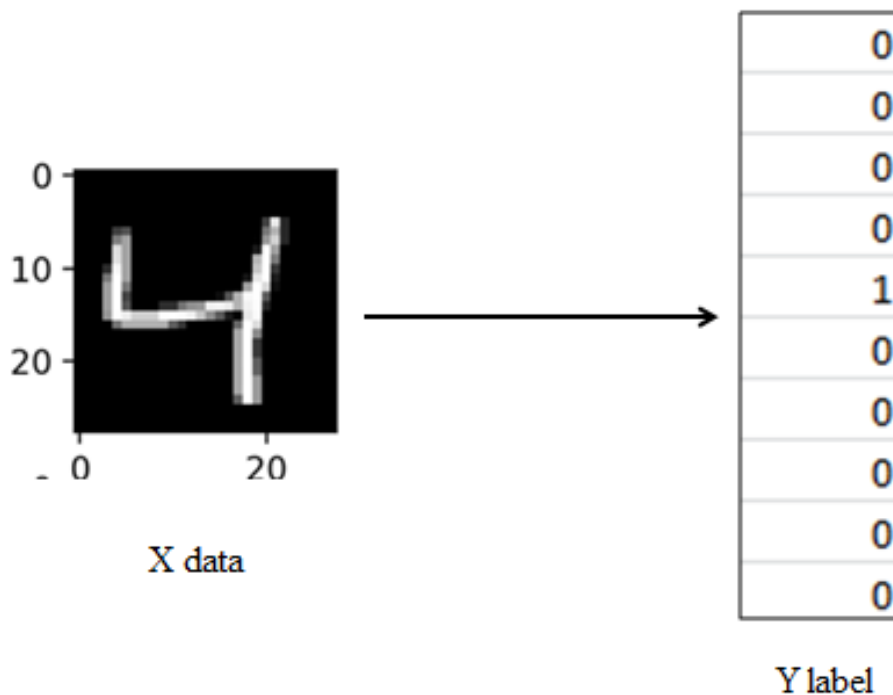


Figure 4. Example of one-hot coding

To define the loss function, we use one-hot encoding which converts the label of the image from a numeric value to a vector of the same size as the model output. For example:

In general, the label of data is the number i is the vector v of size $10 * 1$ with $v_{i+1} = 1$ and other values equal to 0. So with the above convention on percentages, one-hot encoding makes sense that we are 100% sure that this image is number 4.

Now we have the real value (label) in the form of one-hot encoding the predicted value at the output layer after the softmax function of the same size $10 * 1$.

0	a_1
0	a_2
0	a_3
0	a_4
1	a_5
0	a_6
0	a_7
0	a_8
0	a_9
0	a_{10}
y true value	\hat{y} predict value

Figure 5. Example of the real value and the output layer

Desire is a_5 is close to 1 and other values are close to 0 because that means the model can correctly predict the input image is image number 4. We use a cross-entropy loss on the one-hot encoded output as: $L = -\sum_{i=1}^{i=10} y_i * \log(\hat{y}_i)$ [4]

Suppose that the number is 4, then the loss function $L = -\log(\hat{y}_5)$. The L function is small when the value predicts is the closest to the true value and very large when the model predicts wrong. In other words, L is smaller then the predicted model is getting to the true value. So the model problem is problematic to find the smallest value of L . In keras framework the loss function defined above called "categorical_crossentropy".

2.1.2.5. Python code in keras framework

In this section, we use python code [5]. However, in the training phase, we use `batch_size=64`, `epochs=5` and the result is more accurate (approximately 100%):

```
#1. Use the necessary libraries
import numpy as np
import matplotlib.pyplot as plt
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras.utils import np_utils
from keras.datasets import mnist

# 2. Load MNIST data
(X_train, y_train), (X_test, y_test) = mnist.load_data()
X_val, y_val = X_train[50000:60000,:], y_train[50000:60000]
X_train, y_train = X_train[:50000,:], y_train[:50000]

# 3. Reshape data to satisfy the keras requirment
X_train = X_train.reshape(X_train.shape[0], 28, 28, 1)
X_val = X_val.reshape(X_val.shape[0], 28, 28, 1)
X_test = X_test.reshape(X_test.shape[0], 28, 28, 1)

# 4. One hot encoding label (Y)
Y_train = np_utils.to_categorical(y_train, 10)
Y_val = np_utils.to_categorical(y_val, 10)
Y_test = np_utils.to_categorical(y_test, 10)
print(the first value of y ', y_train[0])
print('y value after one-hot encoding ',Y_train[0])

# 5. Difine model
model = Sequential()
    # Add Convolutional layer with 32 kernel, size kernel 3*3
    model.add(Conv2D(32, (3, 3), activation='sigmoid',
input_shape=(28,28,1)))
    # Add Convolutional layer
    model.add(Conv2D(32, (3, 3), activation='sigmoid'))
    # Add Max pooling layer
    model.add(MaxPooling2D(pool_size=(2,2)))
    # Flatten layer
```

```

model.add(Flatten())
# Add Fully Connected layer with 128 nodes and use sigmoid
model.add(Dense(128, activation='sigmoid'))
model.add(Dense(10, activation='softmax'))

# 6. Compile model
model.compile(loss='categorical_crossentropy', optimizer='adam',
              metrics=['accuracy'])

```

7. Train model

```

H = model.fit(X_train, Y_train, validation_data=(X_val, Y_val),
             batch_size=64, epochs=5, verbose=1)

```

The result is very good with an accuracy of about 100% like below:

```

Epoch 1/5
782/782 [=====] - 82s 105ms/step - loss:
7.9526e-04 - accuracy: 0.9999 - val_loss: 0.0424 - val_accuracy:
0.9894

Epoch 2/5
782/782 [=====] - 81s 104ms/step - loss:
1.7001e-04 - accuracy: 1.0000 - val_loss: 0.0417 - val_accuracy:
0.9896

Epoch 3/5
782/782 [=====] - 81s 104ms/step - loss:
1.0931e-04 - accuracy: 1.0000 - val_loss: 0.0421 - val_accuracy:
0.9899

Epoch 4/5
782/782 [=====] - 81s 104ms/step - loss:
8.3609e-05 - accuracy: 1.0000 - val_loss: 0.0429 - val_accuracy:
0.9900

Epoch 5/5
782/782 [=====] - 81s 104ms/step - loss:
6.5229e-05 - accuracy: 1.0000 - val_loss: 0.0432 - val_accuracy:
0.9901

```

8. Graphing the loss function, accuracy of training set and validation set

```

fig = plt.figure()
numOfEpoch = 10

```



```
plt.plot(np.arange(0, numofEpoch), H.history['loss'],
label='training loss')

plt.plot(np.arange(0, numofEpoch), H.history['val_loss'],
label='validation loss')

plt.plot(np.arange(0, numofEpoch), H.history['acc'],
label='accuracy')

plt.plot(np.arange(0, numofEpoch), H.history['val_acc'],
label='validation accuracy')

plt.title('Accuracy and Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss|Accuracy')
plt.legend()

# 9. Validate model with test set
score = model.evaluate(X_test, Y_test, verbose=0) print(score)

# 10. Predict the image
plt.imshow(X_test[0].reshape(28,28), cmap='gray')
y_predict = model.predict(X_test[0].reshape(1,28,28,1))
print('predict value: ', np.argmax(y_predict))
```

3. CONCLUSION

The paper presents a method use CNN model with keras framework for image classification, a popular and useful problem in computer vision field. There are some applications of image classification, for example, diagnosis of X-ray image of patient help to predict cancer, classify and recognize handwritten numbers (letter) which presented in this paper helps to automatically read the license plate, text. The traffic classification signs support for self-driving cars..

Keras framework used to resolve the problem in this paper has some advantages. For example, simplicity, back-end support, pre-trained models, fast experimentation, great community and calibre documentation. And therefore the result achieved is very precise. However, Keras does not support features of dynamic chart creation. The errors given by the Keras library are not effective, it is not very useful and helpful to detect the root cause of the error. The framework is not capable to handle low-level computations. So it focuses on fast experimentation and is chosen by the beginners as well as by the researchers.

The presented model for classifying image is applied for other classification such as classify cars, boats, animals, flowers... and in each applied problem, we change the learning

rate, epoch, batch size value, some layers are added to validate the effectiveness of the model. That is the issue that will be implemented in the near future.

REFERENCE

1. Szeliski R. (2011), “*Computer Vision Algorithms and Applications*”. London: Springer, pp. 1-25.
2. Jeff Hale.(2018), Deep Learning Framework Power Scores 2018—Who’s On Top in Usage, Interest, and Popularity? *Journal of Machine Learning Times*, Available from: <http://www.predictiveanalyticsworld.com/machinelearningtimes/deep-learning-framework-power-scores-2018-whos-on-top-in-usage-interest-and-popularity/9716/>
3. Carlos Affonso, André Luis Debiasio Rossi, Fábio Henrique Antunes Vieira, André Carlos Ponce de Leon Ferreira de Carvalho (2017), “ *Deep learning for biological image classification*” ,Expert Systems with Applications, Volume 85, pp 114-122, ISSN 0957-4174.
4. Christopher M. Bishop.(2006) “*Pattern Recognition and Machine Learning*”. Springer, pp235–240.

KERAS VÀ ỨNG DỤNG CHO BÀI TOÁN PHÂN LOẠI HÌNH ẢNH

Tóm tắt: Keras là một API học sâu viết bằng ngôn ngữ Python, được phát triển với mục tiêu cho phép thử nghiệm nhanh. Bài báo trình bày phương pháp phân loại hình ảnh sử dụng thuật toán CNN với Keras. Mô hình áp dụng đơn giản và cho kết quả có độ chính xác cao. Kết quả nghiên cứu có thể áp dụng để giải quyết tốt các bài toán phân loại hình ảnh khác.

Từ khóa: Thuật toán, thị giác máy tính, CNN, phân loại ảnh, keras, mô hình.