Documentation for Image Classification

Neural Network Project

We introduce image classification project using "tensorflow" and convolutional neural network (CNN) in python.

Convolutional neural networks (CNNs) are a type of neural networks that is used when dealing with image data, audio data or whenever finding patterns in data is required.

Image classification is the process of categorizing and labeling groups of pixels or vectors within an image based on specific rules[1].

In this project, we use a "keras" data set (CIFAR-10). The CIFAR-10 dataset consists of 60000 32x32 color 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 [2]:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 |

We are going to train the neural network to recognize these images then we our going to get some "unseen" images from the internet and feed them into the neural network to evaluate its classification whether it is right or not.

As the data is already inside of the "Keras" the only thing we need to do is to call a load function and then store the data in tuples, training tuple and testing tuple in order to using them. The function will return training and testing data in below format, where images (race of pixels) and labels (such as horse, deer ...).

“(training\_images,training\_labels),(testing\_images,testing\_labels) = datasets.cifar10.load\_data()”

The next step is to normalize the data and scale it down because the pixels are activated from 0 to 255, so the activation depending on how bright the pixel is. We want to have more convenient work with data so that all values are from 0 to 1 by divide it by 255.

“training\_images, testing\_images = training\_images/255, testing\_images/255”

Data now is prepared. Now we are going to define class names list due to that the dataset labels are defined by discrete numbers. Note that order is important. Then we visualize 16 sample of the dataset to see how data set looks like.

“class\_names=['Plane','Car','Bird','Cat','Deer','Dog','Frog','Horse','Ship','Truck'] plt.figure(figsize=(12, 12))

for i in range(16): plt.subplot(4, 4, i+1) plt.xticks([])

plt.yticks([]) plt.imshow((training\_images[i]),cmap=plt.cm.binary) plt.xlabel(class\_names[training\_labels[i][0]])

plt.tight\_layout() plt.show()”

References:

1. https:/[/www.scie](http://www.sciencedirect.com/topics/engineering/image-classification)n[cedirect.com/topics/engineering/image-classification](http://www.sciencedirect.com/topics/engineering/image-classification)
2. <https://www.cs.toronto.edu/~kriz/cifar.html>

**The next step is to build the model used to make the classification .**

**First: Some libraries that help build the model were used, and they are:**

“ from keras.layers import Conv2D, MaxPooling2D,

GlobalAveragePooling2D

from keras.layers import Dropout, Flatten, Dense from keras.models import Sequential

from keras.layers.convolutional import Convolution2D, MaxPooling2D import tensorflow as tf “.

**And then reduce the number of images and labels used for training and testing in order to reduce the time used to do the training and testing part through :**

“training\_images=training\_images[:20000] training\_labels=training\_labels[:20000] testing\_images=testing\_images[:4000] testing\_labels=testing\_labels[:4000]”.

**Then the two-part model was built, part of which is to build the CNN, and the images are prepared to be entered on the NN model so that the classification part is applied through it.**

“model = models.Sequential() model.add(layers.Conv2D(32, (3, 3), activation='relu', input\_shape=(32,32,3))) model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(64,(3, 3), activation='relu'))

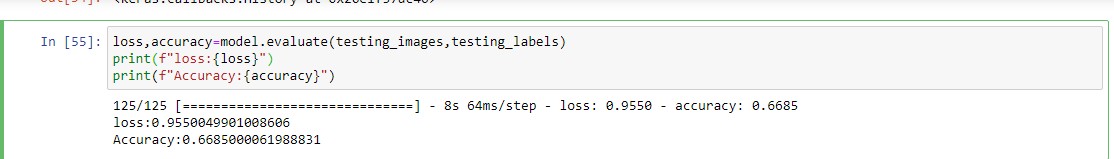
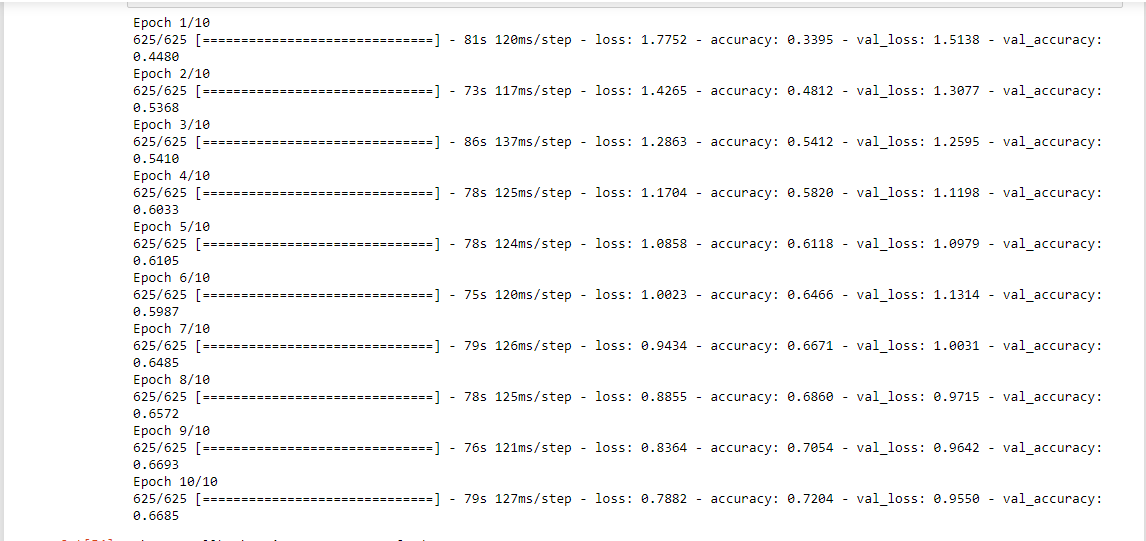
model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(64, (3, 3), activation='relu')) model.add(layers.Dropout(0.2)) model.add(layers.Flatten()) model.add(layers.Dense(64, activation='relu')) model.add(layers.Dense(10, activation='softmax'))”.

**After that, training was done for the images in all the epochs used, and the loss and accuracy were displayed for each epoch through :**

“model.compile(optimizer='adam', loss='sparse\_categorical\_crossentropy', metrics=['accuracy'])

model.fit(training\_images,training\_labels,epochs=10,validation\_data=(tes ting\_images,testing\_labels))”.



**The images were then tested in order to calculate their accuracy. Using:**

“loss,accuracy=model.evaluate(testing\_images,testing\_labels) print(f"loss:{loss}")

print(f"Accuracy:{accuracy}")”

**In this step, the model is downloaded and saved for easy use at any time,as shown in following code.**

“model.save('image\_classifier.model')

model= models.load\_model('image\_classifier.model')”.

**After the model was built ,trained ,tested, and accurate, it will now test its ability to predict and match images ,and this will be done through the following code :**

“from keras.preprocessing.image import img\_to\_array image=cv2.imread(r'C:\Users\user\Documents\project\_nn\car.jpg') image=cv2.cvtColor(image,cv2.COLOR\_BGR2RGB ) plt.imshow(image)

image= img\_to\_array(image) print("NumPy array info:") print(type(image))

image = cv2.resize(image, (32,32)) prediction=model.predict(np.array([image])/255) print("Predictions:")

print(prediction) print(np.argmax(prediction)) index=np.argmax(prediction) print(f'prediction is {class\_names[index]}') “.

**And that picture will show that the prediction was made correctly and the accuracy of prediction is high.**



**The details of the model will be displayed what happened in each layer ,and parameter numbers ,output shape ,all through a function called “summary”**

**,explained by the following code:**

**“model.summary()”**

