

Integrated Experiential Learning



Northeastern University

ALY6080, SPRING 2020

MODULE 8 and 9: HACKATHON WEEK

SUBMITTED BY: ADSAR SHIVANI, KC SANJOG, MISHRA ANUPREETA

EMAIL ID: adsar.s@husky.neu.edu, Kc.s@husky.neu.edu, mishra.anu@husky.neu.edu

NUID: 001399374, 001050752, 001765981

SUBMITTED TO: PROF. MAX MASNICK

DATE: 06/07/2020

Introduction

We have performed Image Classification on the data using deep learning in python programming language. Moreover, we have worked on building a neural network in Keras using python on the CIFAR-10 dataset. Our group consists of three members, Anupreeta Mishra, Shivani Adsar, and Sanjog KC. As we know, Image Classification plays a major role in categorizing the pixels in an image by performing classification on the images(*Image Classification - an Overview | Sciencedirect Topics*, n.d.). The concept of Image Classification has helped us in gaining experience in working on supervised and unsupervised machine learning algorithms. Moreover, we have worked on Python for performing data analysis and predictions on the data, which would improve our python programming skills, thereby, enhancing our professional profile. We have used Tensorflow and Keras, neural network libraries for performing image classification on the data. Nowadays, various companies are working on image classification in the medical field, Colour Processing, and Pattern Recognition. This experience has helped us in gaining practical programming skills on machine learning concepts that are highly required in most of the companies for the positions of a Data Scientist or Data Analyst.

Analysis

Dataset:

The CIFAR-10 is one of the popular datasets for image classification, consisting of around 60000 color images, classified into 10 classes. The dataset is divided into 10 different classes as: Airplane, Car, Bird, Cat, Deer, Dog, Frog, Horse, Ship, and Truck. This dataset is available in the module present in Keras.

Loading the dataset

- Initially, we have loaded the dataset from the Keras datasets module and tried to plot some images from the dataset to visualize the images. Also, we installed the required Keras and TensorFlow packages.
- We know, Keras and Tensorflow are popular frameworks for performing deep learning. The Tensorflow, is an open-source platform used in machine learning that supports easier model building through various levels of abstraction.
- Besides, Keras is a neural network library that runs using the TensorFlow, has a user-friendly, modular, and composable design making it easier to extend.

Neural Network Architecture

- We have used the neural network, so that our model recognizes the patterns in the data, thereby making predictions based on similar patterns.
- Some of the required layers were imported for the creation of the neural network. Further, we have converted the pixel values present in the dataset into float type and then, normalized our dataset to avoid biased results.

Sequential Model

A sequential model is used for a stack of layers where each layer has exactly one input tensor and one output tensor (*Sequential Models in Keras - DeepLearning Frameworks*, n.d.).

We have used the convolutional neural network that classifies images by assigning weights to them. The convolutional layers are responsible for capturing features from the image. Moreover, the pooling layer decreases the convolutional power by the process of dimensionality reduction and therefore extracts the dominant features from the image.

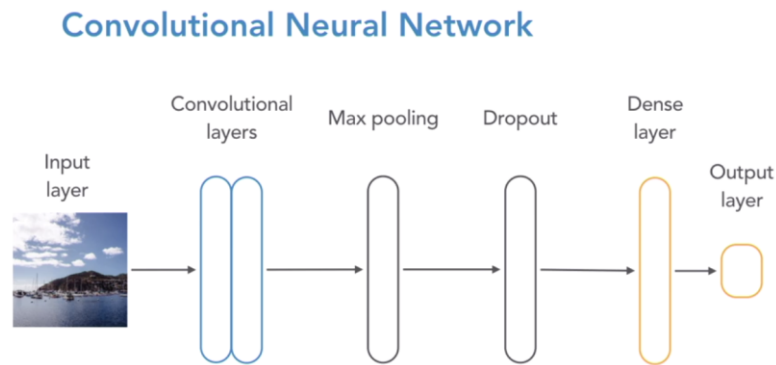


Fig.1: Convolutional Neural Network

We have used max pooling that returns the maximum value from the area of the image covered by the kernel, and suppresses the noise.

```
model = Sequential()
model.add(Conv2D(32, (3, 3), input_shape=(32, 32, 3),
padding='same', activation='relu',
kernel_constraint=maxnorm(3)))
model.add(Dropout(0.2))
model.add(Conv2D(32, (3, 3), activation='relu', padding='same', kernel_constraint=maxnorm(3)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(512, activation='relu', kernel_constraint=maxnorm(3)))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))
```

Fig.2: Sequential Model

We have implemented a sequential model bypassing the list of layers to the sequential constructor. Furthermore, we have configured the optimizer for our model and compiled the same. The summary of the sequential model was analyzed where the input data had assigned weights.

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 32, 32, 32)	896
dropout_1 (Dropout)	(None, 32, 32, 32)	0
conv2d_2 (Conv2D)	(None, 32, 32, 32)	9248
max_pooling2d_1 (MaxPooling2D)	(None, 16, 16, 32)	0
flatten_1 (Flatten)	(None, 8192)	0
dense_1 (Dense)	(None, 512)	4194816
dropout_2 (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 10)	5130

Total params: 4,210,890
Trainable params: 4,210,890
Non-trainable params: 0

Fig.3: Summary of model

The model was trained on the train data and validation was performed on the test data. Moreover, the accuracy of the model was observed on the testing data and our model was saved.

```
sgd = SGD(lr=0.01, momentum=0.9, decay=(0.01 / 25), nesterov=False)

model.compile(loss='categorical_crossentropy',
              optimizer=sgd,
              metrics=['accuracy'])

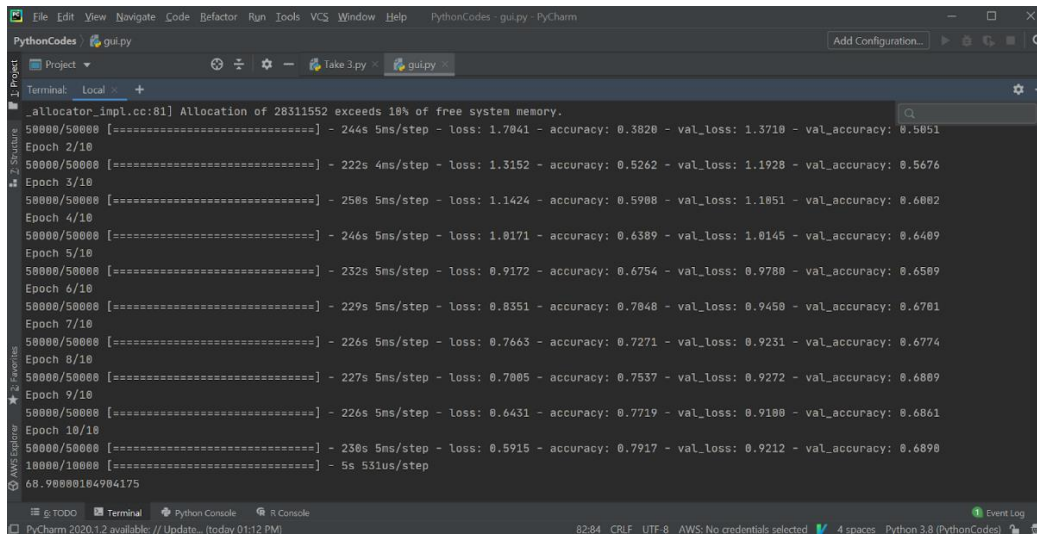
model.summary()

model.fit(train_X, train_Y,
          validation_data=(test_X, test_Y),
          epochs=10, batch_size=32)

_, acc = model.evaluate(test_X, test_Y)
print(acc * 100)

model.save("model1_cifar_10epoch.h5")
```

Fig.4: Model Building



```

PythonCodes - guilpy - PyCharm
PythonCodes / guilpy
Terminal: Local
..._allocator_impl.cc:81] Allocation of 28311552 exceeds 10% of free system memory.
50000/50000 [=====] - 244s 5ms/step - loss: 1.7841 - accuracy: 0.3820 - val_loss: 1.3710 - val_accuracy: 0.5851
Epoch 2/10
50000/50000 [=====] - 222s 4ms/step - loss: 1.3152 - accuracy: 0.5262 - val_loss: 1.1928 - val_accuracy: 0.5676
Epoch 3/10
50000/50000 [=====] - 258s 5ms/step - loss: 1.1424 - accuracy: 0.5908 - val_loss: 1.1851 - val_accuracy: 0.6082
Epoch 4/10
50000/50000 [=====] - 246s 5ms/step - loss: 1.0171 - accuracy: 0.6389 - val_loss: 1.0145 - val_accuracy: 0.6409
Epoch 5/10
50000/50000 [=====] - 232s 5ms/step - loss: 0.9172 - accuracy: 0.6754 - val_loss: 0.9780 - val_accuracy: 0.6589
Epoch 6/10
50000/50000 [=====] - 229s 5ms/step - loss: 0.8351 - accuracy: 0.7048 - val_loss: 0.9450 - val_accuracy: 0.6781
Epoch 7/10
50000/50000 [=====] - 226s 5ms/step - loss: 0.7663 - accuracy: 0.7271 - val_loss: 0.9231 - val_accuracy: 0.6774
Epoch 8/10
50000/50000 [=====] - 227s 5ms/step - loss: 0.7085 - accuracy: 0.7537 - val_loss: 0.9272 - val_accuracy: 0.6889
Epoch 9/10
50000/50000 [=====] - 226s 5ms/step - loss: 0.6431 - accuracy: 0.7719 - val_loss: 0.9188 - val_accuracy: 0.6861
Epoch 10/10
50000/50000 [=====] - 238s 5ms/step - loss: 0.5915 - accuracy: 0.7917 - val_loss: 0.9212 - val_accuracy: 0.6898
10000/10000 [=====] - 5s 531us/step
68.98888104984175

```

Fig.5: Accuracy of the model

Furthermore, we have used a dictionary to map the output classes and make predictions from the model.

Image Classification Graphical User Interface

We have used the ‘Tkinter’ package which is a GUI toolkit provided in python. Moreover, we have loaded the trained model for classifying the images. Also, the dictionary to label the CIFAR-10 dataset classes has been imported and the GUI was initialized.

As we can view the output of our model below, the image has been classified correctly as an automobile.

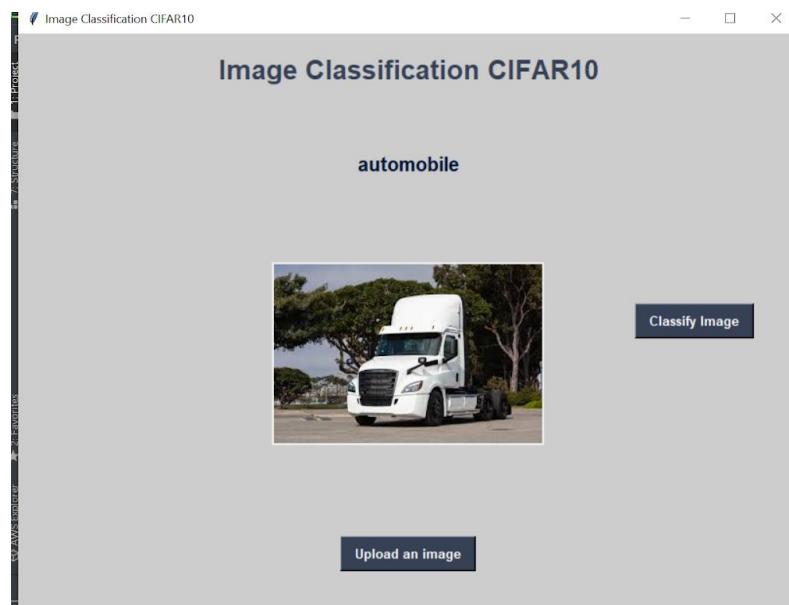


Fig.6: Output 1

Similarly, the below image has been classified correctly as a horse.



Fig.7: Output 2

Challenges faced while working on the data:

- The model has classified some images incorrectly due to the presence of other features that dominated the image. For example:
The image of a Truck was misclassified as an airplane. As we can see, the major part of the image consists of the sky and therefore the model classifies the image incorrectly as an airplane instead of a truck. This is due to the presence of background noise, hence, we would be working on reducing the noise and improving the accuracy of our model by enhancing our algorithm using optimization techniques.

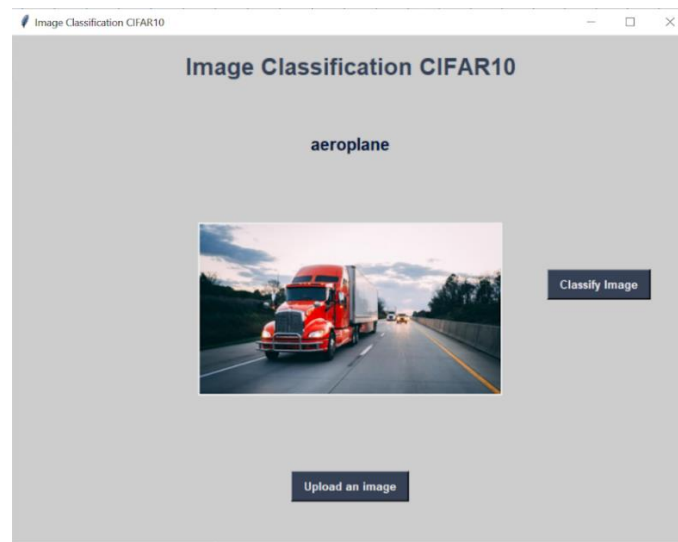


Fig.8: Output 3

- Due to COVID-19, we were not able to conduct face to face meetings, and solving each other's issues through virtual meetings can be time-consuming. Also, it was difficult to resolve technical errors through virtual meetings, especially when the team member lacked technical expertise. To overcome the technicalities, we conducted frequent virtual meetings.
- Also, we have used Google Docs for sharing documents for live editing. Moreover, we have faced challenges while working on improving the accuracy of our model. We have used online resources related to machine learning and image processing while working on the project to overcome technical issues.
- The concept of Image Classification is vast and required some time to get familiarized with the working of the algorithm. Hence, the project had detailed research involved for which we accessed online resources to help each other in understanding technical concepts.

Accessing Success

The results of our image classifier model have helped in determining the success of our analysis. We have ensured correct image classification by our model, which has determined its accurate working. However, there is a slight improvement in accuracy which is required for our model, as it has classified 3% of the images incorrectly. We intend on improving the predictions of our model through optimized techniques. Furthermore, through our collaborated efforts, we have determined the success of our model by validating our data on various algorithms to gain the highest accuracy. Besides, we have worked on enhancing and increasing the compatibility of our classifier for better predictions. Also, we had distributed the work equally amongst us for a better success rate of our project. Furthermore, we performed initial exploratory data analysis to get familiarized with the data and then progressed towards performing in-depth analysis through deep learning algorithm, this systematic approach helped in achieving the correct outcome.

Applications

Some of the applications of image classifications are as follows:

- Gaming: Gaming companies and Virtual reality use image classification to enhance the experience of gamers and enhance the gaming experiences. This has helped companies to increase the revenue in the global market.
- Social Media: Social media Facebook has added a feature for visually impaired people by combining facial recognition techniques and automatic text technologies to generate an accurate description of the photo content as well as describing who exactly is in the photo without being tagged.
- Security: FaceID is widely used for security purposes. Many apps are created that use the FaceID approach to lock and unlock their home and their property.

Conclusion

The project on Image Classification helped in gaining practical experience in deep learning. This has helped in improving our skillsets in learning TensorFlow, Keras, and deep learning algorithms using python. We have used neural networks for implementing image classification for our dataset, which has used 10 classes of the dataset in the Keras module to predict the type of object in the image.

References

- [1] Image Classification. (n.d.). Retrieved May 26, 2020, from <https://www.sciencedirect.com/topics/computer-science/image-classification>
- [2] Sequential models in Keras - DeepLearning Frameworks. (n.d.). Retrieved from <https://www.coursera.org/lecture/ai/sequential-models-in-keras-RBbLP>