AIT – Budapest

Fall 2021

Deep Learning

Final Project:

Deep Learning CIFAR-10 (with dropout, batch normalization and data augmentation)

Supervisor: **Dr. Balint Gyires-Toth**

Student name: **Nedal Ahmad**

Introduction:

Deep learning has significantly improved image recognition systems which can be applied in every aspect of our lives. In this project, I will build a Deep Neural Network (DNN) to recognize certain objects in some photo using CIFAR-10 dataset to predict one of ten classes of objects. adjust state of the art DNN for recognition of other new-objects but we don't need to do training again.

In this project, I will model a classification task with the CIFAR10 dataset using Keras.

Previous solutions:

Firstly, I used PyCharm v. 2018.2 with Microsoft Visual C++ 2015 Redistributable update 3

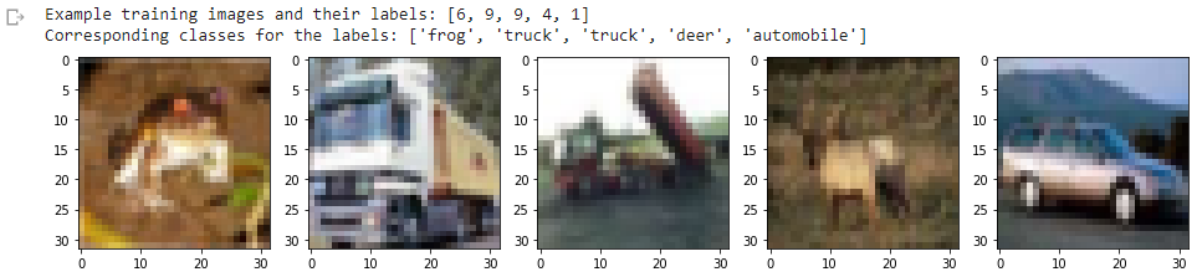
that needs a very specific version of dependencies and libraries and not the smoothest way to develop my solution as pure Jupyter Notebook or google colab!

Dataset:

We have 50000 training and 10000 test images in the dataset. The images have a structure of (32,32,3) which correspond to (width, height, RGB color channels).

CIFAR10 http://www.cs.toronto.edu/~kriz/cifar.html

For each image there is a corresponding label, which is a class index.



Proposed method:

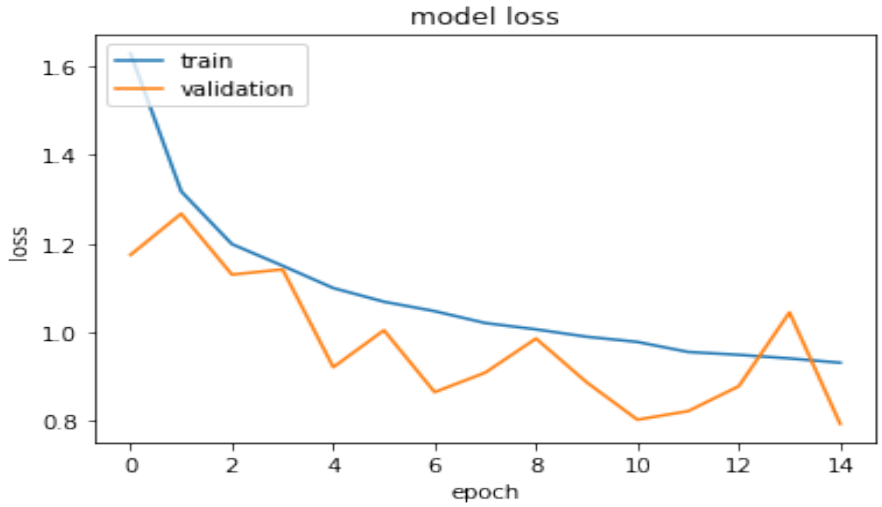
I will use deep learning to build CNN model

1- Loading the dataset
2- Examining the dataset
3- Preparing the dataset
4- Creating CNN model
5- Add dropout, batch normalization and data augmentation

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 30, 30, 32)        896

 conv2d_1 (Conv2D)           (None, 28, 28, 32)        9248

 batch_normalization (BatchN  (None, 28, 28, 32)       128
 ormalization)

 activation (Activation)     (None, 28, 28, 32)        0

 max_pooling2d (MaxPooling2D  (None, 14, 14, 32)       0
 )

 dropout (Dropout)           (None, 14, 14, 32)        0

 flatten (Flatten)           (None, 6272)              0

 dense (Dense)               (None, 256)               1605888

 batch_normalization_1 (Batc  (None, 256)              1024
 hNormalization)

 activation_1 (Activation)   (None, 256)               0

 dropout_1 (Dropout)         (None, 256)               0

 dense_1 (Dense)             (None, 10)                2570

=================================================================
```

6- Fits the model on batches with real-time data augmentation then start training



Evaluation method:

First with the “model.evaluate” command to compare the x-test input data and the y-test output:

```
[ ]  score = model.evaluate(X_test, y_test, batch_size=128, verbose=0)
```

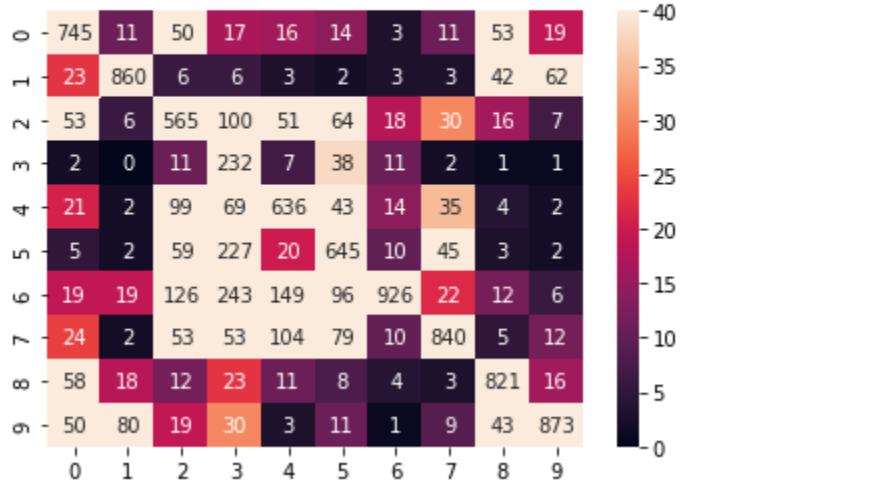
```
[ ]  print(model.metrics_names)
     print(score)
```

```
['loss', 'accuracy']
[0.8028714060783386, 0.7142999768257141]
```

Then showing the confusion matrix available with seaborn library that will point to the high values in its diagonal axis:

```
[ ]  sns.heatmap(conf, annot=True, fmt="d", vmax=40)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ffa63587510>
```



Results and discussion:

1- By using Convolutional Neural Networks, we can take advantage of the special structure of the inputs. Convolutions are translation invariant, and this makes them especially well-suited for processing images.

2- In most cases, larger models have a tendency to overfit training data. While getting good performance on the training set, they will perform poorly on the test set. Regularization methods are used to prevent overfitting, making these larger models generalize better:

   a- Dropout: masking a random subset of its outputs (zeroing them) for every input with probability p and scaling up the rest of the outputs by 1/(1 - p).

   b- Batch normalization: works by normalizing layer outputs to a running mean and variance. This will speed up training and improves the final performance of the model. The running statistics are fixed at test time.

   c- Data Augmentation: used to increase the amount of data by adding slightly modified copies of already existing data or newly created synthetic data from existing data. It acts as a regularizes and helps reduce overfitting when training a machine learning model.

3- In my case the size of the dataset is large enough and no need for augmentation

4- The solution can be enhanced, and the accuracy may improve by:

   a- We can Introduce pretrained networks like Inception V3, and apply transfer learning.

   b- Fine tune the model and change hyperparameters.