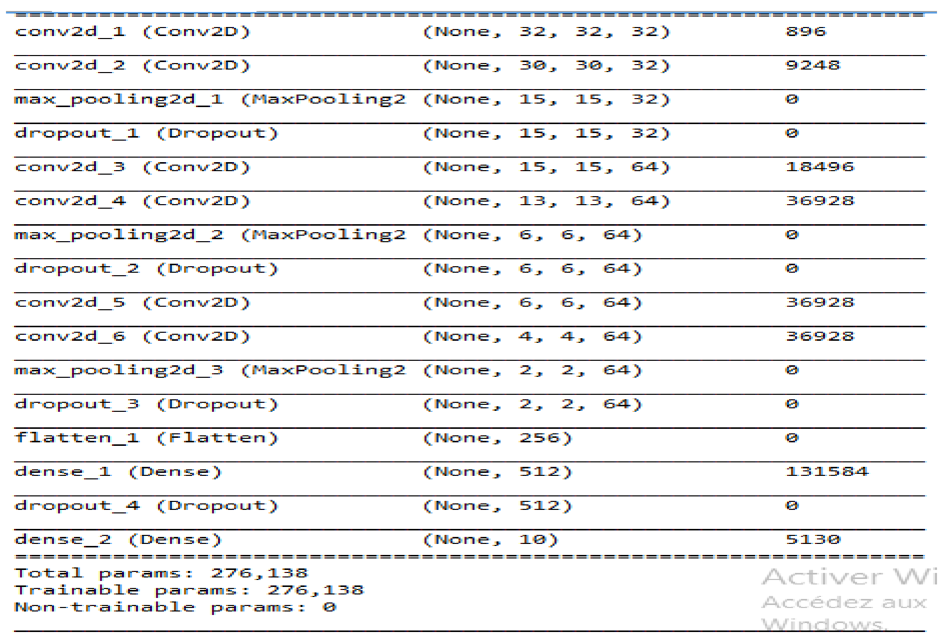
Project: Image Classification using Convolution Neural Networks (Keras CNN-CIFAR-10)

1-Overview: In this project I tried to improve the accuracy by using a CNN model and image augmentations in **Keras**. Build a neural network which will classify the images by itself by looking at the image it is fed. I worked with the CIFAR-10 dataset which contains images different kinds of objects. These objects are labeled as - airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. These images are of size 32 x 32 with three channels red, green, and blue. The dataset is broken up into testing and training data. The testing data contains 10,000 images whereas the training data contains 50,000 images.

2-Step of project: (The explication of steps for the second and third networks it's similar to the first network just I changed the size-batch and number epochs and others parameters)

1. Imports: import data and constant (names class and number of class)

2. The Network (model): For implementing a CNN, I will stack up Convolution Layers, followed by Max Pooling layers. Also include Dropout to avoid over fitting. Finally, we will add a Dense or fully connected layer followed by a softmax layer. In this code, I used 6 convolution layers and 3 fully-connected layer. Adds convolution layers with 32 filters / kernels with a window size of 3×3. And 64 filter with the same size of window . Add a max pooling layer with window size 2×2. Add a dropout layer with a dropout ratio of 0.25. In the final lines, add the dense layer which performs the classification among 10 classes using a softmax layer. In the model summary we can see the shapes and parameters of each layer:



```
=================================================================
conv2d_1 (Conv2D)            (None, 32, 32, 32)        896
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 30, 30, 32)        9248
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 15, 15, 32)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 15, 15, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 15, 15, 64)        18496
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 13, 13, 64)        36928
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 6, 6, 64)          0
_________________________________________________________________
dropout_2 (Dropout)          (None, 6, 6, 64)          0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 6, 6, 64)          36928
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 4, 4, 64)          36928
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 2, 2, 64)          0
_________________________________________________________________
dropout_3 (Dropout)          (None, 2, 2, 64)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               131584
_________________________________________________________________
dropout_4 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 10)                5130
=================================================================
Total params: 276,138
Trainable params: 276,138
Non-trainable params: 0
_________________________________________________________________
```

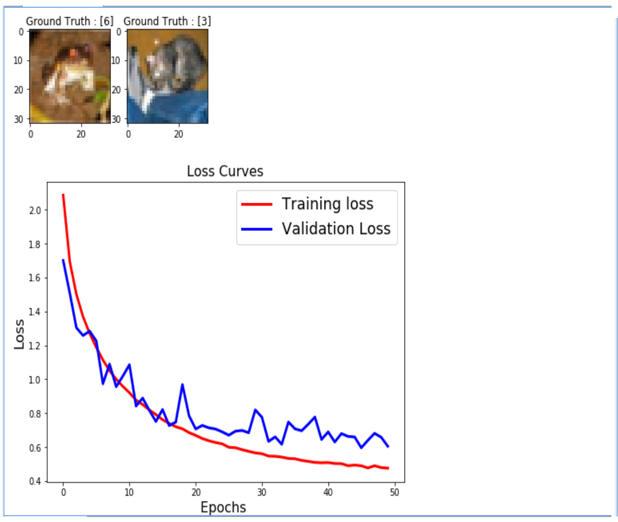
Used padding in the first layer, the output shape is same as the input (32×32). But the second conv layer shrinks by 2 pixels in both dimensions. Also, the output size after pooling layer decreases by half since we have used a stride of 2 and a window size of 2×2. The final dropout layer has an output of 2x2x64. This has to be converted to a single array. The flatten layer which converts the 3D array into a 1D array of size 2x2x64 = 256. The final layer has 10 nodes since there are 10 classes.

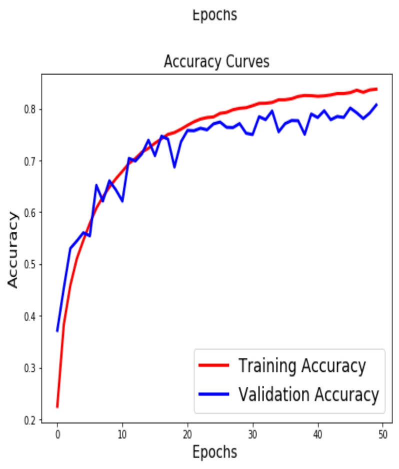
3. Training the network

For training the network, I followed the simple workflow of create "compile fit"   Since it is a 10 class classification problem, used a categorical cross entropy loss (objective of model is minimize it) and use "RMSProp" optimizer to train the network. Metrics for any classification problem we will want to set this to metrics= [accuracy].In this project code run for 50 epochs.

4. Loss & Accuracy Curves:

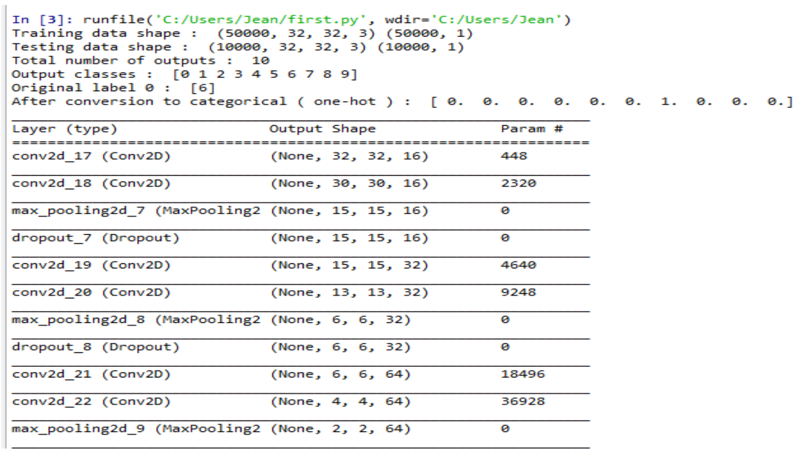
1. First network :



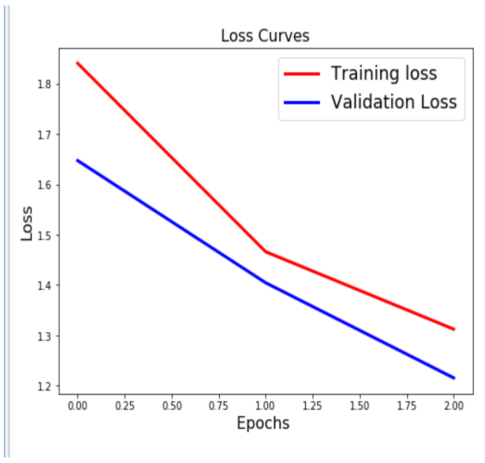


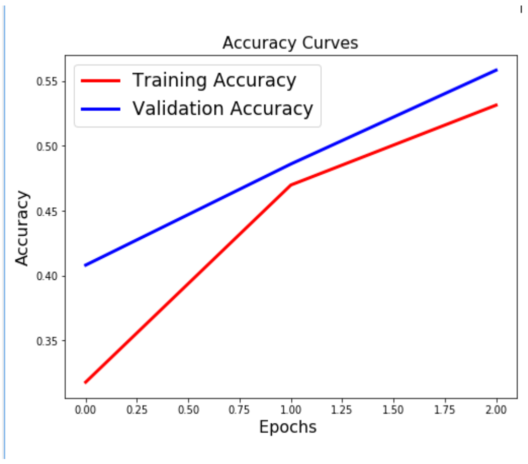
```
In [2]: runfile('C:/Users/Jean/cifare_data.py', wdir='C:/Users/Jean')ERROR: execution aborted
```

2. Second network:

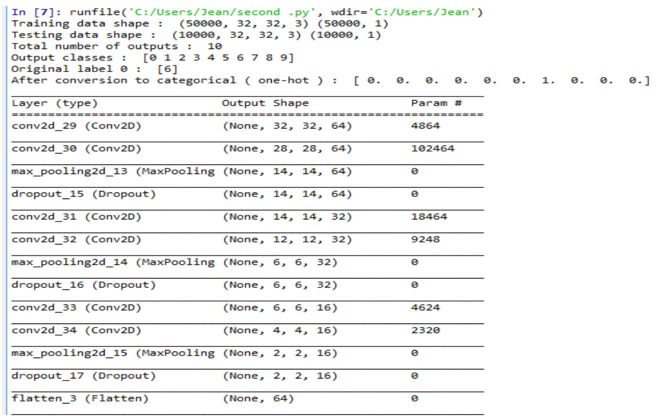


```
In [3]: runfile('C:/Users/Jean/first.py', wdir='C:/Users/Jean')
Training data shape :  (50000, 32, 32, 3) (50000, 1)
Testing data shape :  (10000, 32, 32, 3) (10000, 1)
Total number of outputs :  10
Output classes :  [0 1 2 3 4 5 6 7 8 9]
Original label 0 :  [6]
After conversion to categorical ( one-hot ) :  [ 0.  0.  0.  0.  0.  0.  1.  0.  0.  0.]
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_17 (Conv2D)           (None, 32, 32, 16)        448
_________________________________________________________________
conv2d_18 (Conv2D)           (None, 30, 30, 16)        2320
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 15, 15, 16)        0
_________________________________________________________________
dropout_7 (Dropout)          (None, 15, 15, 16)        0
_________________________________________________________________
conv2d_19 (Conv2D)           (None, 15, 15, 32)        4640
_________________________________________________________________
conv2d_20 (Conv2D)           (None, 13, 13, 32)        9248
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 6, 6, 32)          0
_________________________________________________________________
dropout_8 (Dropout)          (None, 6, 6, 32)          0
_________________________________________________________________
conv2d_21 (Conv2D)           (None, 6, 6, 64)          18496
_________________________________________________________________
conv2d_22 (Conv2D)           (None, 4, 4, 64)          36928
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 2, 2, 64)          0
_________________________________________________________________
```

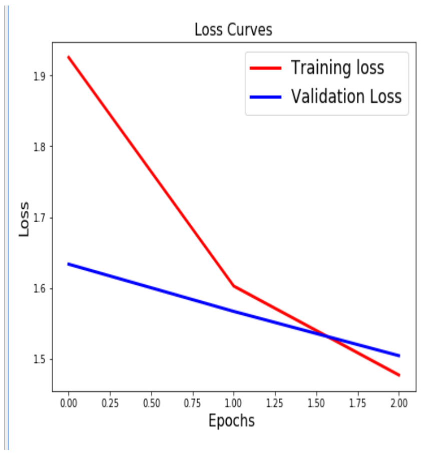


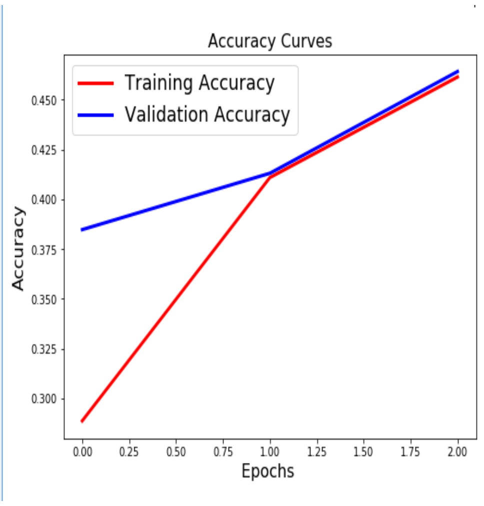


3. Third network:



```
In [7]: runfile('C:/Users/Jean/second .py', wdir='C:/Users/Jean')
Training data shape :  (50000, 32, 32, 3) (50000, 1)
Testing data shape :  (10000, 32, 32, 3) (10000, 1)
Total number of outputs :  10
Output classes :  [0 1 2 3 4 5 6 7 8 9]
Original label 0 :  [6]
After conversion to categorical ( one-hot ) :  [ 0.  0.  0.  0.  0.  0.  1.  0.  0.  0.]
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_29 (Conv2D)           (None, 32, 32, 64)        4864
_________________________________________________________________
conv2d_30 (Conv2D)           (None, 28, 28, 64)        102464
_________________________________________________________________
max_pooling2d_13 (MaxPooling (None, 14, 14, 64)        0
_________________________________________________________________
dropout_15 (Dropout)         (None, 14, 14, 64)        0
_________________________________________________________________
conv2d_31 (Conv2D)           (None, 14, 14, 32)        18464
_________________________________________________________________
conv2d_32 (Conv2D)           (None, 12, 12, 32)        9248
_________________________________________________________________
max_pooling2d_14 (MaxPooling (None, 6, 6, 32)          0
_________________________________________________________________
dropout_16 (Dropout)         (None, 6, 6, 32)          0
_________________________________________________________________
conv2d_33 (Conv2D)           (None, 6, 6, 16)          4624
_________________________________________________________________
conv2d_34 (Conv2D)           (None, 4, 4, 16)          2320
_________________________________________________________________
max_pooling2d_15 (MaxPooling (None, 2, 2, 16)          0
_________________________________________________________________
dropout_17 (Dropout)         (None, 2, 2, 16)          0
_________________________________________________________________
flatten_3 (Flatten)          (None, 64)                0
_________________________________________________________________
```





From the above curves, we can see there is a considerable difference between **the training and validation loss**. This is a sign of Overfitting. Already I used Dropout in the network for avoid it , then why is it still overfitting. Its possible reduce overfitting by using Data **Augmentation** for get better accuracy.

Using Data Augmentation

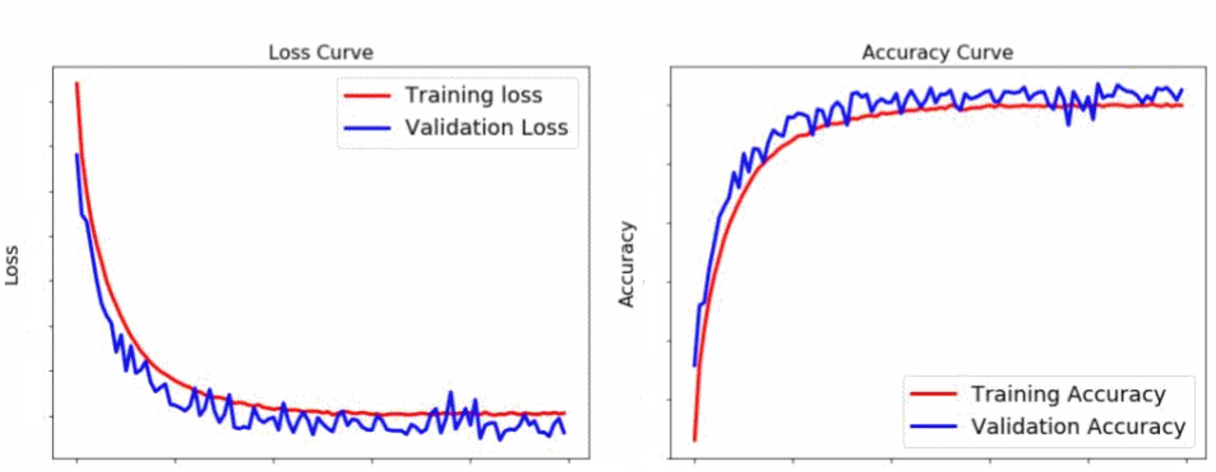
Using Data Augmentation: It is the process of artificially creating more images from the images you already have by changing the size, orientation etc of the image. It can be a tedious task but fortunately, this can be done in Keras using the **Image Data Generator** instance.

I have provided some of the operations that can be done using the Image Data Generator for data augmentation. This includes rotation of the image, shifting the image left/right/top/bottom by some amount, flip the image horizontally or vertically.

2. Training with Data Augmentation: Similar to the previous section, create the model, but use data augmentation while training. I used ImageDataGenerator for creating a generator which feed the network.

1. Create the model and configure it.
2. Create an ImageDataGenerator object and configure it using parameters for horizontal flip, and image translation.
3. The datagen.flow() function generates batches of data, after performing the data transformations / augmentation specified during the instantiation of the data generator.
4. The fit_generator function will train the model using the data obtained in batches from the datagen.flow function.

3.Loss & Accuracy Curves(First network)



The test accuracy is greater than training accuracy. This means that the model has generalized very well. This comes from the fact that the model has been trained on much worse data so it is finding the normal test data easier to classify.

Visualize 10 pictures that the network fails to classify:

I would like to visualize miss classification but unfortunately doesn’t work the last step of my code .