Object Recognition

The objective of this lab is very simple, to recognize objects in images. You will be working with a well-known dataset called CIFAR-10.

You can learn more about this dataset and download it here:

https://www.cs.toronto.edu/~kriz/cifar.html (https://www.cs.toronto.edu/~kriz/cifar.html)

In the webpage above, they also included a few publications based on CIFAR-10 data, which showed some amazing accuracies. The worst network on the page (a shallow convolutional neural network) can classify images with rouhgly 75% accuracy.

1. Write a function to load data

The dataset webpage in the previous section also provide a simple way to load data from your harddrive using pickle. You may use their function for this exercise.

Construct two numpy arrays for train images and train labels from data_batch_1 to data_batch_5. Then, construct two numpy arrays for test images, and test labels from test batch file. The original image size is 32 x 32 x 3. You may flatten the arrays so the final arrays are of size 1 x 3072.

Load data

จาก 6 directory มี ข้อมูล dir ละ 10000 ซึ่งเป็นข้อมูลประเภท dictionary จึงได้สร้าง list ไว้ใช้เก็บข้อมูล 4 list คือ list ไว้สำหรับเก็บ training data, testing data, label of training data, label of testing data

In [1]:
```
import numpy as np
```

In [2]:
```
def unpickle(file):
    import pickle
    with open(file, 'rb') as fo:
        dict = pickle.load(fo, encoding='bytes')
    return dict
```

In [3]:
```
directory = [r"C:\Users\dell\Desktop\module8&9\AI\Lab03\cifar-10-batches-py\data_
        r"C:\Users\dell\Desktop\module8&9\AI\Lab03\cifar-10-batches-py\data_batch
        r"C:\Users\dell\Desktop\module8&9\AI\Lab03\cifar-10-batches-py\data_batch
        r"C:\Users\dell\Desktop\module8&9\AI\Lab03\cifar-10-batches-py\data_batch
        r"C:\Users\dell\Desktop\module8&9\AI\Lab03\cifar-10-batches-py\data_batch
        r"C:\Users\dell\Desktop\module8&9\AI\Lab03\cifar-10-batches-py\test_batch
```

In [4]:
```
data_train = []
label_train = []
data_test = []
label_test =[]
```

In [5]:
```
for i in range(len(directory)):
    if i != len(directory)-1:
        raw_data = unpickle(directory[i])
        for ind in range(len(raw_data[b'data'])):
            data_train.append(raw_data[b'data'][ind])
            label_train.append(raw_data[b'labels'][ind])
    else:
        raw_data = unpickle(directory[i])
        for ind in range(len(raw_data[b'data'])):
            data_test.append(raw_data[b'data'][ind])
            label_test.append(raw_data[b'labels'][ind])
```

2. Classify Dogs v.s. Cats

Let's start simple by creating logistic regression model to classify images. We will select only two classes of images for this exercise.

1. From 50,000 train images and 10,000 test images, we want to reduce the data size. Write code to filter only dog images (label = 3) and cat images (label = 5).
2. Create a logistic regression model to classify cats and dogs. Report your accuracy.

Reduce data size

อาศัย loop เพื่อ select data ที่มี label = 3 or 5

In [6]:
```
fil_data_train = []
fil_data_test =[]
fil_label_train = []
fil_label_test = []

for i in range(len(data_train)):
    if label_train[i] == 3:
        fil_data_train.append(data_train[i])
        fil_label_train.append(3)
    elif label_train[i] == 5:
        fil_data_train.append(data_train[i])
        fil_label_train.append(5)

for i in range(len(data_test)):
    if label_test[i] == 3:
        fil_data_test.append(data_test[i])
        fil_label_test.append(3)
    elif label_test[i] == 5:
        fil_data_test.append(data_test[i])
        fil_label_test.append(5)
```

Create Logistic Regression Model

ได้ accuracy = 53.5 %

In [7]:
```
from sklearn.linear_model import LogisticRegression
```

In [8]:
```
logregr = LogisticRegression()
```

In [9]:
```
logregr.fit(fil_data_train,fil_label_train)
```

Out[9]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False)
```

In [10]:
```
acc = logregr.score(fil_data_test,fil_label_test)
print('---------------Logistic Regression model--------------------\n')
print('           The accuracy is '+str(acc)+' or '+str(acc*100)+'%'+'\n')
print('------------------------------------------------------------')
```

```
---------------Logistic Regression model--------------------

           The accuracy is 0.5325 or 53.25%

------------------------------------------------------------
```

3. The Real Challenge

The majority of your score for this lab will come from this real challenge. You are going to construct a neural network model to classify 10 classes of images from CIFAR-10 dataset. You will get half the credits for this one if you complete the assignment, and will get another half if you can exceed the target accuracy of 75%. (You may use any combination of sklearn, opencv, or tensorflow to do this exercise).

Design at least 3 variants of neural network models. Each model should have different architectures. (Do not vary just a few parameters, the architecture of the network must change in each model). In your notebook, explain your experiments in details and display the accuracy score for each experiment.

1. MLP Classifier

```
1st Experiment process with preprocessing data
    MLPClassifier(activation='logistic', alpha=0.0001, batch_size='auto',
       beta_1=0.9, beta_2=0.999, early_stopping=True, epsilon=1e-08,
       hidden_layer_sizes=(50, 50, 50, 50, 50), learning_rate='constant',
       learning_rate_init=0.001, max_iter=500, momentum=0.9,
       nesterovs_momentum=True, power_t=0.5, random_state=None,
       shuffle=True, solver='adam', tol=0.0001, validation_fraction=0.1,
       verbose=True, warm_start=True)
    accuracy = 0.3718
------------------------------------------------------------------------
2nd Experiment process with preprocessing data
    MLPClassifier(activation='logistic', alpha=0.0001, batch_size='auto',
       beta_1=0.9, beta_2=0.999, early_stopping=True, epsilon=1e-08,
       hidden_layer_sizes=(100, 100, 100, 100, 100),
       learning_rate='constant', learning_rate_init=0.001, max_iter=500,
       momentum=0.9, nesterovs_momentum=True, power_t=0.5,
       random_state=None, shuffle=True, solver='adam', tol=0.0001,
       validation_fraction=0.1, verbose=True, warm_start=True)
    accuracy = 0.422
------------------------------------------------------------------------
3rd Experiment process with preprocessing data
    MLPClassifier(activation='logistic', alpha=0.0001, batch_size='auto',
       beta_1=0.9, beta_2=0.999, early_stopping=True, epsilon=1e-08,
       hidden_layer_sizes=(500, 400, 300, 100, 100),
       learning_rate='adaptive', learning_rate_init=0.001, max_iter=500,
       momentum=0.9, nesterovs_momentum=True, power_t=0.5,
       random_state=None, shuffle=True, solver='adam', tol=0.0001,
       validation_fraction=0.1, verbose=True, warm_start=True)
    accuracy = 0.4637
------------------------------------------------------------------------
4th Experiment process with preprocessing data
    MLPClassifier(activation='logistic', alpha=0.0001, batch_size='auto',
       beta_1=0.9, beta_2=0.999, early_stopping=True, epsilon=1e-08,
       hidden_layer_sizes=(1000, 500, 400), learning_rate='adaptive',
       learning_rate_init=0.001, max_iter=500, momentum=0.9,
       nesterovs_momentum=True, power_t=0.5, random_state=None,
       shuffle=True, solver='adam', tol=0.0001, validation_fraction=0.1,
       verbose=True, warm_start=True)
    accuracy = 0.4976
------------------------------------------------------------------------
```

```
5th Experiment *****process with unpreprocessing data
    MLP = MLPClassifier(activation='logistic', alpha=0.0001, batch_size='auto',
       beta_1=0.9, beta_2=0.999, early_stopping=True, epsilon=1e-08,
       hidden_layer_sizes=(1000, 500, 400), learning_rate='adaptive',
       learning_rate_init=0.001, max_iter=500, momentum=0.9,
       nesterovs_momentum=True, power_t=0.5, random_state=None,
       shuffle=True, solver='adam', tol=0.0001, validation_fraction=0.1,
       verbose=True, warm_start=True)
    accuracy = 0.137
--------------------------------------------------------------------------
6th Experiment ******process with unpreprocessing data
    MLP = MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto',
       beta_1=0.9, beta_2=0.999, early_stopping=True, epsilon=1e-08,
       hidden_layer_sizes=(100,100,100,100,100,100,100,100,100,100),
learning_rate='adaptive',
       learning_rate_init=0.001, max_iter=500, momentum=0.9,
       nesterovs_momentum=True, power_t=0.5, random_state=None,
       shuffle=True, solver='adam', tol=0.0001, validation_fraction=0.1,
       verbose=True, warm_start=True)
    accuracy = 0.4434
--------------------------------------------------------------------------
7th Experiment process with preprocessing data
    MLP = MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto',
       beta_1=0.9, beta_2=0.999, early_stopping=True, epsilon=1e-08,
       hidden_layer_sizes=(100,100,100,100,100,100,100,100,100,100),
learning_rate='adaptive',
       learning_rate_init=0.001, max_iter=500, momentum=0.9,
       nesterovs_momentum=True, power_t=0.5, random_state=None,
       shuffle=True, solver='adam', tol=0.0001, validation_fraction=0.1,
       verbose=True, warm_start=True)
    accuracy = 0.5108
--------------------------------------------------------------------------
8th Experiment process with preprocessing data
    MLP = MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto',
       beta_1=0.9, beta_2=0.999, early_stopping=True, epsilon=1e-08,
       hidden_layer_sizes=(500,400,400,300,300,100,100,100,100,100),
learning_rate='adaptive',
       learning_rate_init=0.001, max_iter=500, momentum=0.9,
       nesterovs_momentum=True, power_t=0.5, random_state=None,
       shuffle=True, solver='adam', tol=0.0001, validation_fraction=0.1,
       verbose=True, warm_start=True)
    accuracy = 0.5409
--------------------------------------------------------------------------
9th Experiment process with preprocessing data
    MLP = MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto',
       beta_1=0.9, beta_2=0.999, early_stopping=True, epsilon=1e-08,
       hidden_layer_sizes=(1000,900,800,700,600,500,400,300,200,100),
learning_rate='adaptive',
       learning_rate_init=0.001, max_iter=500, momentum=0.9,
       nesterovs_momentum=True, power_t=0.5, random_state=None,
       shuffle=True, solver='adam', tol=0.0001, validation_fraction=0.1,
       verbose=True, warm_start=True)
    accuracy = 0.5378
--------------------------------------------------------------------------
10th Experiment process with preprocessing data
    MLP = MLPClassifier(activation='tanh', alpha=0.0001, batch_size='auto',
       beta_1=0.9, beta_2=0.999, early_stopping=True, epsilon=1e-08,
```

```
       hidden_layer_sizes=(1000,900,800,700,600,500,400,300,200,100),
learning_rate='adaptive',
       learning_rate_init=0.001, max_iter=500, momentum=0.9,
       nesterovs_momentum=True, power_t=0.5, random_state=None,
       shuffle=True, solver='adam', tol=0.0001, validation_fraction=0.1,
       verbose=True, warm_start=True)
    accuracy = 0.3678
--------------------------------------------------------------------------
11th Experiment process with preprocessing data
    MLP = MLPClassifier(activation='logistic', alpha=0.0001, batch_size='auto',
       beta_1=0.9, beta_2=0.999, early_stopping=True, epsilon=1e-08,
       hidden_layer_sizes=(1000,900,800,700,600,500,400,300,200,100),
learning_rate='adaptive',
       learning_rate_init=0.001, max_iter=500, momentum=0.9,
       nesterovs_momentum=True, power_t=0.5, random_state=None,
       shuffle=True, solver='adam', tol=0.0001, validation_fraction=0.1,
       verbose=True, warm_start=True)
    accuracy = 0.10
```

Preprocessing

In [11]:
```
from sklearn.preprocessing import StandardScaler
```

In [12]:
```
scaler = StandardScaler()
scaler.fit(data_train)
x_train = scaler.transform(data_train)
x_test = scaler.transform(data_test)
y_train = label_train
y_test = label_test
```

Create Model

In [13]:
```
from sklearn.neural_network import MLPClassifier
```

In [14]:
```
MLP = MLPClassifier(activation='logistic', alpha=0.0001, batch_size='auto',
       beta_1=0.9, beta_2=0.999, early_stopping=True, epsilon=1e-08,
       hidden_layer_sizes=(50, 50, 50, 50, 50), learning_rate='constant',
       learning_rate_init=0.001, max_iter=500, momentum=0.9,
       nesterovs_momentum=True, power_t=0.5, random_state=None,
       shuffle=True, solver='adam', tol=0.0001, validation_fraction=0.1,
       verbose=True, warm_start=True)
MLP.fit(x_train,y_train)
acc = MLP.score(x_test,y_test)
print('---------------1st MLP Classifier model--------------------\n')
print('------------Process with preprocessing data----------------\n')
print('           The accuracy is '+str(acc)+' or '+str(acc*100)+'%'+'\n')
print('-------------------------------------------------------------')
```

```
Iteration 1, loss = 2.22528966
Validation score: 0.185800
Iteration 2, loss = 2.04818440
Validation score: 0.192800
Iteration 3, loss = 2.01528579
Validation score: 0.204800
Iteration 4, loss = 1.99214377
Validation score: 0.222800
Iteration 5, loss = 1.97523115
Validation score: 0.211400
Iteration 6, loss = 1.95733482
Validation score: 0.230000
Iteration 7, loss = 1.93876435
Validation score: 0.238600
Iteration 8, loss = 1.92170168
Validation score: 0.237000
Iteration 9, loss = 1.90699760
Validation score: 0.249400
Iteration 10, loss = 1.88843833
Validation score: 0.272200
Iteration 11, loss = 1.85692325
Validation score: 0.302200
Iteration 12, loss = 1.81587432
Validation score: 0.296000
Iteration 13, loss = 1.78822993
Validation score: 0.295400
Iteration 14, loss = 1.76797491
Validation score: 0.304800
Iteration 15, loss = 1.75132663
Validation score: 0.308000
Iteration 16, loss = 1.73504445
Validation score: 0.317600
Iteration 17, loss = 1.72146247
Validation score: 0.317200
Iteration 18, loss = 1.70636475
Validation score: 0.328600
Iteration 19, loss = 1.69112348
Validation score: 0.323600
Iteration 20, loss = 1.67491058
Validation score: 0.329800
Iteration 21, loss = 1.65928564
Validation score: 0.332400
```

```
Iteration 22, loss = 1.64651128
Validation score: 0.333400
Iteration 23, loss = 1.63293108
Validation score: 0.340000
Iteration 24, loss = 1.61832550
Validation score: 0.346200
Iteration 25, loss = 1.60028884
Validation score: 0.348200
Iteration 26, loss = 1.58746013
Validation score: 0.348200
Iteration 27, loss = 1.57339816
Validation score: 0.349000
Iteration 28, loss = 1.55848785
Validation score: 0.355000
Iteration 29, loss = 1.54443941
Validation score: 0.354800
Iteration 30, loss = 1.53136336
Validation score: 0.360600
Iteration 31, loss = 1.52208363
Validation score: 0.364400
Iteration 32, loss = 1.50302751
Validation score: 0.365600
Iteration 33, loss = 1.49370847
Validation score: 0.368800
Iteration 34, loss = 1.47918908
Validation score: 0.367400
Iteration 35, loss = 1.46968031
Validation score: 0.370200
Iteration 36, loss = 1.45653573
Validation score: 0.374000
Iteration 37, loss = 1.44752093
Validation score: 0.365400
Iteration 38, loss = 1.43582211
Validation score: 0.373600
Iteration 39, loss = 1.42750397
Validation score: 0.373200
Validation score did not improve more than tol=0.000100 for two consecutive epo
chs. Stopping.
---------------1st MLP Classifier model--------------------

------------Process with preprocessing data----------------

           The accuracy is 0.3718 or 37.18%

-------------------------------------------------------------
```

In [15]:
```
MLP = MLPClassifier(activation='logistic', alpha=0.0001, batch_size='auto',
       beta_1=0.9, beta_2=0.999, early_stopping=True, epsilon=1e-08,
       hidden_layer_sizes=(100, 100, 100, 100, 100),
       learning_rate='constant', learning_rate_init=0.001, max_iter=500,
       momentum=0.9, nesterovs_momentum=True, power_t=0.5,
       random_state=None, shuffle=True, solver='adam', tol=0.0001,
       validation_fraction=0.1, verbose=True, warm_start=True)
MLP.fit(x_train,y_train)
acc = MLP.score(x_test,y_test)
print('---------------2nd MLP Classifier model--------------------\n')
print('------------Process with preprocessing data----------------\n')
print('           The accuracy is '+str(acc)+' or '+str(acc*100)+'%'+'\n')
print('-------------------------------------------------------------')
```

```
Iteration 1, loss = 2.15456693
Validation score: 0.190200
Iteration 2, loss = 2.02686805
Validation score: 0.211400
Iteration 3, loss = 1.99216041
Validation score: 0.203600
Iteration 4, loss = 1.96443496
Validation score: 0.222800
Iteration 5, loss = 1.93909555
Validation score: 0.219000
Iteration 6, loss = 1.91196981
Validation score: 0.239800
Iteration 7, loss = 1.87619862
Validation score: 0.267000
Iteration 8, loss = 1.80776114
Validation score: 0.307000
Iteration 9, loss = 1.74241435
Validation score: 0.330400
Iteration 10, loss = 1.70395315
Validation score: 0.349800
Iteration 11, loss = 1.67125003
Validation score: 0.362800
Iteration 12, loss = 1.63488125
Validation score: 0.372200
Iteration 13, loss = 1.60394036
Validation score: 0.363600
Iteration 14, loss = 1.57814230
Validation score: 0.380000
Iteration 15, loss = 1.54870427
Validation score: 0.383400
Iteration 16, loss = 1.52320927
Validation score: 0.389800
Iteration 17, loss = 1.49373081
Validation score: 0.395000
Iteration 18, loss = 1.46875613
Validation score: 0.393800
Iteration 19, loss = 1.43655718
Validation score: 0.404000
Iteration 20, loss = 1.41656561
Validation score: 0.400000
Iteration 21, loss = 1.38578918
Validation score: 0.400800
```

```
Iteration 22, loss = 1.36432092
Validation score: 0.407400
Iteration 23, loss = 1.34521627
Validation score: 0.404600
Iteration 24, loss = 1.32180484
Validation score: 0.412400
Iteration 25, loss = 1.30045529
Validation score: 0.415800
Iteration 26, loss = 1.28296756
Validation score: 0.415800
Iteration 27, loss = 1.25923137
Validation score: 0.411000
Iteration 28, loss = 1.24256792
Validation score: 0.418000
Iteration 29, loss = 1.22737721
Validation score: 0.420200
Iteration 30, loss = 1.20861236
Validation score: 0.411600
Iteration 31, loss = 1.19281382
Validation score: 0.416400
Iteration 32, loss = 1.18103914
Validation score: 0.411600
Validation score did not improve more than tol=0.000100 for two consecutive epo
chs. Stopping.
---------------2nd MLP Classifier model--------------------

------------Process with preprocessing data----------------

           The accuracy is 0.422 or 42.199999999999996%

-----------------------------------------------------------
```

In [16]:
```
MLP = MLPClassifier(activation='logistic', alpha=0.0001, batch_size='auto',
       beta_1=0.9, beta_2=0.999, early_stopping=True, epsilon=1e-08,
       hidden_layer_sizes=(500, 400, 300, 100, 100),
       learning_rate='adaptive', learning_rate_init=0.001, max_iter=500,
       momentum=0.9, nesterovs_momentum=True, power_t=0.5,
       random_state=None, shuffle=True, solver='adam', tol=0.0001,
       validation_fraction=0.1, verbose=True, warm_start=True)
MLP.fit(x_train,y_train)
acc = MLP.score(x_test,y_test)
print('---------------3rd MLP Classifier model--------------------\n')
print('------------Process with preprocessing data----------------\n')
print('           The accuracy is '+str(acc)+' or '+str(acc*100)+'%'+'\n')
print('------------------------------------------------------------')
```

```
Iteration 1, loss = 2.10604733
Validation score: 0.202000
Iteration 2, loss = 1.96744914
Validation score: 0.277000
Iteration 3, loss = 1.81963945
Validation score: 0.335600
Iteration 4, loss = 1.70779112
Validation score: 0.382000
Iteration 5, loss = 1.60825782
Validation score: 0.423200
Iteration 6, loss = 1.53457109
Validation score: 0.434000
Iteration 7, loss = 1.47103216
Validation score: 0.448600
Iteration 8, loss = 1.41457515
Validation score: 0.455400
Iteration 9, loss = 1.37016463
Validation score: 0.444000
Iteration 10, loss = 1.31575987
Validation score: 0.452400
Iteration 11, loss = 1.27075653
Validation score: 0.458000
Iteration 12, loss = 1.23139332
Validation score: 0.468200
Iteration 13, loss = 1.18179987
Validation score: 0.461600
Iteration 14, loss = 1.14386309
Validation score: 0.467800
Iteration 15, loss = 1.08775994
Validation score: 0.464800
Validation score did not improve more than tol=0.000100 for two consecutive epo
chs. Stopping.
---------------3rd MLP Classifier model--------------------

------------Process with preprocessing data----------------

           The accuracy is 0.4637 or 46.37%

------------------------------------------------------------
```

In [17]:
```
MLP = MLPClassifier(activation='logistic', alpha=0.0001, batch_size='auto',
       beta_1=0.9, beta_2=0.999, early_stopping=True, epsilon=1e-08,
       hidden_layer_sizes=(1000, 500, 400), learning_rate='adaptive',
       learning_rate_init=0.001, max_iter=500, momentum=0.9,
       nesterovs_momentum=True, power_t=0.5, random_state=None,
       shuffle=True, solver='adam', tol=0.0001, validation_fraction=0.1,
       verbose=True, warm_start=True)
MLP.fit(x_train,y_train)
acc = MLP.score(x_test,y_test)
print('---------------4th MLP Classifier model--------------------\n')
print('------------Process with preprocessing data----------------\n')
print('           The accuracy is '+str(acc)+' or '+str(acc*100)+'%'+'\n')
print('------------------------------------------------------------')
```

```
Iteration 1, loss = 1.81586586
Validation score: 0.406600
Iteration 2, loss = 1.57970449
Validation score: 0.444400
Iteration 3, loss = 1.47506294
Validation score: 0.465200
Iteration 4, loss = 1.38908820
Validation score: 0.472400
Iteration 5, loss = 1.32161253
Validation score: 0.484800
Iteration 6, loss = 1.25433929
Validation score: 0.498400
Iteration 7, loss = 1.19023244
Validation score: 0.499200
Iteration 8, loss = 1.13519231
Validation score: 0.494800
Iteration 9, loss = 1.07372844
Validation score: 0.494400
Iteration 10, loss = 1.01196842
Validation score: 0.504000
Iteration 11, loss = 0.95024389
Validation score: 0.509600
Iteration 12, loss = 0.88645396
Validation score: 0.496600
Iteration 13, loss = 0.83951156
Validation score: 0.514800
Iteration 14, loss = 0.77272687
Validation score: 0.504800
Iteration 15, loss = 0.72634928
Validation score: 0.510600
Iteration 16, loss = 0.66815233
Validation score: 0.500400
Validation score did not improve more than tol=0.000100 for two consecutive epo
chs. Stopping.
---------------4th MLP Classifier model--------------------

------------Process with preprocessing data----------------

           The accuracy is 0.4976 or 49.76%

------------------------------------------------------------
```

In [18]:
```
MLP = MLPClassifier(activation='logistic', alpha=0.0001, batch_size='auto',
       beta_1=0.9, beta_2=0.999, early_stopping=True, epsilon=1e-08,
       hidden_layer_sizes=(1000, 500, 400), learning_rate='adaptive',
       learning_rate_init=0.001, max_iter=500, momentum=0.9,
       nesterovs_momentum=True, power_t=0.5, random_state=None,
       shuffle=True, solver='adam', tol=0.0001, validation_fraction=0.1,
       verbose=True, warm_start=True)
MLP.fit(data_train,label_train)
acc = MLP.score(data_test,label_test)
print('---------------5th MLP Classifier model--------------------\n')
print('-----------Process with unpreprocessing data---------------\n')
print('           The accuracy is '+str(acc)+' or '+str(acc*100)+'%'+'\n')
print('-------------------------------------------------------------')
```

```
Iteration 1, loss = 2.31127638
Validation score: 0.095800
Iteration 2, loss = 2.28365547
Validation score: 0.122400
Iteration 3, loss = 2.26960654
Validation score: 0.137200
Iteration 4, loss = 2.23806878
Validation score: 0.146200
Iteration 5, loss = 2.30855971
Validation score: 0.098600
Iteration 6, loss = 2.30452269
Validation score: 0.098400
Iteration 7, loss = 2.30286019
Validation score: 0.096400
Validation score did not improve more than tol=0.000100 for two consecutive epo
chs. Stopping.
---------------5th MLP Classifier model--------------------

-----------Process with unpreprocessing data---------------

           The accuracy is 0.137 or 13.700000000000001%

-------------------------------------------------------------
```

In [19]:
```
MLP = MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto',
       beta_1=0.9, beta_2=0.999, early_stopping=True, epsilon=1e-08,
       hidden_layer_sizes=(100,100,100,100,100,100,100,100,100,100), learning_rat
       learning_rate_init=0.001, max_iter=500, momentum=0.9,
       nesterovs_momentum=True, power_t=0.5, random_state=None,
       shuffle=True, solver='adam', tol=0.0001, validation_fraction=0.1,
       verbose=True, warm_start=True)
MLP.fit(data_train,label_train)
acc = MLP.score(data_test,label_test)
print('---------------6th MLP Classifier model--------------------\n')
print('-----------Process with unpreprocessing data---------------\n')
print('           The accuracy is '+str(acc)+' or '+str(acc*100)+'%'+'\n')
print('-------------------------------------------------------------')
```

```
Iteration 1, loss = 2.66754084
Validation score: 0.297000
Iteration 2, loss = 1.85065128
Validation score: 0.354400
Iteration 3, loss = 1.77459145
Validation score: 0.366000
Iteration 4, loss = 1.72842975
Validation score: 0.355000
Iteration 5, loss = 1.69669273
Validation score: 0.405800
Iteration 6, loss = 1.64747330
Validation score: 0.414600
Iteration 7, loss = 1.63327315
Validation score: 0.401800
Iteration 8, loss = 1.61226285
Validation score: 0.415600
Iteration 9, loss = 1.60499469
Validation score: 0.416200
Iteration 10, loss = 1.58060749
Validation score: 0.431200
Iteration 11, loss = 1.56423379
Validation score: 0.415800
Iteration 12, loss = 1.54568369
Validation score: 0.438600
Iteration 13, loss = 1.53886377
Validation score: 0.445400
Iteration 14, loss = 1.52537428
Validation score: 0.442400
Iteration 15, loss = 1.51633013
Validation score: 0.440000
Iteration 16, loss = 1.52015414
Validation score: 0.433400
Validation score did not improve more than tol=0.000100 for two consecutive epo
chs. Stopping.
---------------6th MLP Classifier model--------------------

-----------Process with unpreprocessing data---------------

           The accuracy is 0.4434 or 44.34%

-------------------------------------------------------------
```

In [20]:
```
MLP = MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto',
       beta_1=0.9, beta_2=0.999, early_stopping=True, epsilon=1e-08,
       hidden_layer_sizes=(100,100,100,100,100,100,100,100,100,100), learning_rat
       learning_rate_init=0.001, max_iter=500, momentum=0.9,
       nesterovs_momentum=True, power_t=0.5, random_state=None,
       shuffle=True, solver='adam', tol=0.0001, validation_fraction=0.1,
       verbose=True, warm_start=True)
MLP.fit(x_train,y_train)
acc = MLP.score(x_test,y_test)
print('---------------7th MLP Classifier model--------------------\n')
print('-----------Process with preprocessing data---------------\n')
print('           The accuracy is '+str(acc)+' or '+str(acc*100)+'%'+'\n')
print('------------------------------------------------------------')
```

```
Iteration 1, loss = 1.79312558
Validation score: 0.397800
Iteration 2, loss = 1.55285671
Validation score: 0.437400
Iteration 3, loss = 1.44625073
Validation score: 0.453800
Iteration 4, loss = 1.37499698
Validation score: 0.461200
Iteration 5, loss = 1.31543335
Validation score: 0.475400
Iteration 6, loss = 1.26160755
Validation score: 0.468800
Iteration 7, loss = 1.21996080
Validation score: 0.483800
Iteration 8, loss = 1.17214318
Validation score: 0.490600
Iteration 9, loss = 1.13189771
Validation score: 0.491200
Iteration 10, loss = 1.09873682
Validation score: 0.498000
Iteration 11, loss = 1.05323321
Validation score: 0.486600
Iteration 12, loss = 1.02254937
Validation score: 0.498800
Iteration 13, loss = 0.99073988
Validation score: 0.502000
Iteration 14, loss = 0.96277519
Validation score: 0.486400
Iteration 15, loss = 0.92139270
Validation score: 0.498800
Iteration 16, loss = 0.90447135
Validation score: 0.498000
Validation score did not improve more than tol=0.000100 for two consecutive epo
chs. Stopping.
---------------7th MLP Classifier model--------------------

-----------Process with preprocessing data---------------

           The accuracy is 0.5108 or 51.080000000000005%

------------------------------------------------------------
```

In [21]:
```
MLP = MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto',
       beta_1=0.9, beta_2=0.999, early_stopping=True, epsilon=1e-08,
       hidden_layer_sizes=(500,400,400,300,300,100,100,100,100,100), learning_rat
       learning_rate_init=0.001, max_iter=500, momentum=0.9,
       nesterovs_momentum=True, power_t=0.5, random_state=None,
       shuffle=True, solver='adam', tol=0.0001, validation_fraction=0.1,
       verbose=True, warm_start=True)
MLP.fit(x_train,y_train)
acc = MLP.score(x_test,y_test)
print('---------------8th MLP Classifier model--------------------\n')
print('-----------Process with preprocessing data---------------\n')
print('           The accuracy is '+str(acc)+' or '+str(acc*100)+'%'+'\n')
print('------------------------------------------------------------')
```

```
Iteration 1, loss = 1.77874073
Validation score: 0.436400
Iteration 2, loss = 1.53629391
Validation score: 0.457400
Iteration 3, loss = 1.41672509
Validation score: 0.490400
Iteration 4, loss = 1.31534418
Validation score: 0.512200
Iteration 5, loss = 1.23427576
Validation score: 0.518800
Iteration 6, loss = 1.15868143
Validation score: 0.513000
Iteration 7, loss = 1.08329536
Validation score: 0.539400
Iteration 8, loss = 1.00615204
Validation score: 0.534400
Iteration 9, loss = 0.93310391
Validation score: 0.531600
Iteration 10, loss = 0.87589228
Validation score: 0.550800
Iteration 11, loss = 0.81383054
Validation score: 0.534600
Iteration 12, loss = 0.74110071
Validation score: 0.543200
Iteration 13, loss = 0.68592842
Validation score: 0.529600
Validation score did not improve more than tol=0.000100 for two consecutive epo
chs. Stopping.
---------------8th MLP Classifier model--------------------

-----------Process with preprocessing data---------------

           The accuracy is 0.5409 or 54.09%

------------------------------------------------------------
```

In [22]:
```
MLP = MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto',
       beta_1=0.9, beta_2=0.999, early_stopping=True, epsilon=1e-08,
       hidden_layer_sizes=(1000,900,800,700,600,500,400,300,200,100), learning_ra
       learning_rate_init=0.001, max_iter=500, momentum=0.9,
       nesterovs_momentum=True, power_t=0.5, random_state=None,
       shuffle=True, solver='adam', tol=0.0001, validation_fraction=0.1,
       verbose=True, warm_start=True)
MLP.fit(x_train,y_train)
acc = MLP.score(x_test,y_test)
print('---------------9th MLP Classifier model--------------------\n')
print('-----------Process with preprocessing data---------------\n')
print('           The accuracy is '+str(acc)+' or '+str(acc*100)+'%'+'\n')
print('------------------------------------------------------------')
```

```
Iteration 1, loss = 1.79064328
Validation score: 0.395400
Iteration 2, loss = 1.55611775
Validation score: 0.433800
Iteration 3, loss = 1.44342729
Validation score: 0.478600
Iteration 4, loss = 1.35078849
Validation score: 0.488200
Iteration 5, loss = 1.26957589
Validation score: 0.494000
Iteration 6, loss = 1.19471808
Validation score: 0.507000
Iteration 7, loss = 1.12021668
Validation score: 0.516800
Iteration 8, loss = 1.04362510
Validation score: 0.511800
Iteration 9, loss = 0.98441128
Validation score: 0.522400
Iteration 10, loss = 0.91210777
Validation score: 0.529600
Iteration 11, loss = 0.83622486
Validation score: 0.527600
Iteration 12, loss = 0.78213183
Validation score: 0.533000
Iteration 13, loss = 0.70677238
Validation score: 0.532000
Iteration 14, loss = 0.65296245
Validation score: 0.525600
Iteration 15, loss = 0.59084766
Validation score: 0.523800
Validation score did not improve more than tol=0.000100 for two consecutive epo
chs. Stopping.
---------------9th MLP Classifier model--------------------

-----------Process with preprocessing data---------------

           The accuracy is 0.5378 or 53.779999999999994%

------------------------------------------------------------
```

In [23]:
```
MLP = MLPClassifier(activation='tanh', alpha=0.0001, batch_size='auto',
       beta_1=0.9, beta_2=0.999, early_stopping=True, epsilon=1e-08,
       hidden_layer_sizes=(1000,900,800,700,600,500,400,300,200,100), learning_ra
       learning_rate_init=0.001, max_iter=500, momentum=0.9,
       nesterovs_momentum=True, power_t=0.5, random_state=None,
       shuffle=True, solver='adam', tol=0.0001, validation_fraction=0.1,
       verbose=True, warm_start=True)
MLP.fit(x_train,y_train)
acc = MLP.score(x_test,y_test)
print('---------------10th MLP Classifier model--------------------\n')
print('-----------Process with preprocessing data------------------\n')
print('           The accuracy is '+str(acc)+' or '+str(acc*100)+'%'+'\n')
print('------------------------------------------------------------')
```

```
Iteration 1, loss = 1.97887843
Validation score: 0.308000
Iteration 2, loss = 1.87301104
Validation score: 0.325000
Iteration 3, loss = 1.82311427
Validation score: 0.340200
Iteration 4, loss = 1.79957459
Validation score: 0.353800
Iteration 5, loss = 1.80377888
Validation score: 0.324200
Iteration 6, loss = 1.84983993
Validation score: 0.328600
Iteration 7, loss = 1.82090762
Validation score: 0.322800
Validation score did not improve more than tol=0.000100 for two consecutive epo
chs. Stopping.
---------------10th MLP Classifier model--------------------

-----------Process with preprocessing data------------------

           The accuracy is 0.3678 or 36.78%

------------------------------------------------------------
```

In [24]:
```
MLP = MLPClassifier(activation='logistic', alpha=0.0001, batch_size='auto',
       beta_1=0.9, beta_2=0.999, early_stopping=True, epsilon=1e-08,
       hidden_layer_sizes=(1000,900,800,700,600,500,400,300,200,100), learning_ra
       learning_rate_init=0.001, max_iter=500, momentum=0.9,
       nesterovs_momentum=True, power_t=0.5, random_state=None,
       shuffle=True, solver='adam', tol=0.0001, validation_fraction=0.1,
       verbose=True, warm_start=True)
MLP.fit(x_train,y_train)
acc = MLP.score(x_test,y_test)
print('---------------11th MLP Classifier model--------------------\n')
print('-----------Process with preprocessing data------------------\n')
print('           The accuracy is '+str(acc)+' or '+str(acc*100)+'%'+'\n')
print('------------------------------------------------------------')
```

```
Iteration 1, loss = 2.30792319
Validation score: 0.096600
Iteration 2, loss = 2.30564063
Validation score: 0.092800
Iteration 3, loss = 2.30499634
Validation score: 0.103800
Iteration 4, loss = 2.30506988
Validation score: 0.096600
Iteration 5, loss = 2.30450753
Validation score: 0.102400
Iteration 6, loss = 2.30420944
Validation score: 0.108000
Iteration 7, loss = 2.30362340
Validation score: 0.092800
Iteration 8, loss = 2.30336573
Validation score: 0.092800
Iteration 9, loss = 2.30314584
Validation score: 0.103800
Validation score did not improve more than tol=0.000100 for two consecutive epo
chs. Stopping.
---------------11th MLP Classifier model--------------------

-----------Process with preprocessing data------------------

           The accuracy is 0.1 or 10.0%

------------------------------------------------------------



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

2. Convolution Neural Network(CNN)

In [29]:
```
from keras.models import Sequential
from keras.layers import Dense,Activation,Flatten,Conv2D,BatchNormalization,Dropo
```

Preprocessing

In [31]:
```
from keras.utils import to_categorical
```

In [50]:
```
x3d_train = np.array(data_train).reshape(np.array(data_train).shape[0],3,32,32)
x3d_test = np.array(data_test).reshape(np.array(data_test).shape[0],3,32,32)

y_train = to_categorical(label_train,++10)
y_test = to_categorical(label_test,10)
```

Model1

In [46]:
```
Model1 = Sequential()

Model1.add(Conv2D(filters=32,kernel_size=(5,5),padding='same',data_format='channe
Model1.add(BatchNormalization(axis=1))
Model1.add(MaxPooling2D(pool_size=(2,2)))

Model1.add(Flatten())
Model1.add(Dense(1000,activation='relu'))
Model1.add(Dense(500,activation='relu'))
Model1.add(Dense(10,activation='softmax'))

Model1.compile(loss='categorical_crossentropy', optimizer='adam',metrics=['accura

Model1.fit(x3d_train,y_train, validation_data=(x3d_test,y_test),epochs = 10,batch_
```

```
Train on 50000 samples, validate on 10000 samples
Epoch 1/10
50000/50000 [==============================] - 1240s 25ms/step - loss: 1.5923 -
acc: 0.4529 - val_loss: 1.6117 - val_acc: 0.4622
Epoch 2/10
50000/50000 [==============================] - 1177s 24ms/step - loss: 1.1611 -
acc: 0.5926 - val_loss: 1.3985 - val_acc: 0.5372
Epoch 3/10
50000/50000 [==============================] - 1183s 24ms/step - loss: 0.9683 -
acc: 0.6643 - val_loss: 1.1417 - val_acc: 0.6163
Epoch 4/10
50000/50000 [==============================] - 1165s 23ms/step - loss: 0.7799 -
acc: 0.7291 - val_loss: 1.1928 - val_acc: 0.6195
Epoch 5/10
50000/50000 [==============================] - 1154s 23ms/step - loss: 0.6118 -
acc: 0.7883 - val_loss: 1.5194 - val_acc: 0.5885
Epoch 6/10
50000/50000 [==============================] - 1141s 23ms/step - loss: 0.4714 -
acc: 0.8394 - val_loss: 1.5636 - val_acc: 0.6117
Epoch 7/10
50000/50000 [==============================] - 1137s 23ms/step - loss: 0.3648 -
acc: 0.8762 - val_loss: 1.4994 - val_acc: 0.6267
Epoch 8/10
50000/50000 [==============================] - 1149s 23ms/step - loss: 0.2959 -
acc: 0.9035 - val_loss: 1.6680 - val_acc: 0.6299
Epoch 9/10
50000/50000 [==============================] - 1157s 23ms/step - loss: 0.2371 -
acc: 0.9230 - val_loss: 1.9181 - val_acc: 0.6199
Epoch 10/10
50000/50000 [==============================] - 1160s 23ms/step - loss: 0.1988 -
acc: 0.9360 - val_loss: 2.1887 - val_acc: 0.5937
```

Out[46]: <keras.callbacks.History at 0x272b03976a0>

```
Architecture:
Conv2D 32 filter(5,5) --> BatchNormalize --> Maxpooling --> Flatten -->
Dense(1000) --> Dense(500) --> Dense(10)
Accuracy = 0.5937 = 59.37%

Result
```

```
Train on 50000 samples, validate on 10000 samples
Epoch 1/10
50000/50000 [==============================] - 1240s 25ms/step - loss: 1.5923 -
acc: 0.4529 - val_loss: 1.6117 - val_acc: 0.4622
Epoch 2/10
50000/50000 [==============================] - 1177s 24ms/step - loss: 1.1611 -
acc: 0.5926 - val_loss: 1.3985 - val_acc: 0.5372
Epoch 3/10
50000/50000 [==============================] - 1183s 24ms/step - loss: 0.9683 -
acc: 0.6643 - val_loss: 1.1417 - val_acc: 0.6163
Epoch 4/10
50000/50000 [==============================] - 1165s 23ms/step - loss: 0.7799 -
acc: 0.7291 - val_loss: 1.1928 - val_acc: 0.6195
Epoch 5/10
50000/50000 [==============================] - 1154s 23ms/step - loss: 0.6118 -
acc: 0.7883 - val_loss: 1.5194 - val_acc: 0.5885
Epoch 6/10
50000/50000 [==============================] - 1141s 23ms/step - loss: 0.4714 -
acc: 0.8394 - val_loss: 1.5636 - val_acc: 0.6117
Epoch 7/10
50000/50000 [==============================] - 1137s 23ms/step - loss: 0.3648 -
acc: 0.8762 - val_loss: 1.4994 - val_acc: 0.6267
Epoch 8/10
50000/50000 [==============================] - 1149s 23ms/step - loss: 0.2959 -
acc: 0.9035 - val_loss: 1.6680 - val_acc: 0.6299
Epoch 9/10
50000/50000 [==============================] - 1157s 23ms/step - loss: 0.2371 -
acc: 0.9230 - val_loss: 1.9181 - val_acc: 0.6199
Epoch 10/10
50000/50000 [==============================] - 1160s 23ms/step - loss: 0.1988 -
acc: 0.9360 - val_loss: 2.1887 - val_acc: 0.5937
```

จากผลลัพธ์ พบว่า model มีความ overfit สูงมาก ทั้งนี้คิดว่าเป็นเพราะมี fully connetor ซ้อนกัน 3 ชั้น และมีจำนวนว node เยอะเกินไป จึงได้ปรับเปลี่ยนเป็น model3 โดย ได้ทำการลด node ของ fully contected layer ลง และทำการเพิ่ม convolution layer เนื่องจากคิดว่ารูปมีทั้งหมด 10 class การที่มี conv เยอะๆจะช่วยให้สามารถแยก class ได้เยอะ และได้ทำการเพิ่ม drop out เพื่อป้องกันการ overfit ของ model

Model3

In [54]:
```
Model3 = Sequential()

Model3.add(Conv2D(filters=128,kernel_size=(8,8),padding='same',data_format='chann
Model3.add(BatchNormalization(axis=1))
Model3.add(Conv2D(filters=64,kernel_size=(4,4),padding='same',data_format='channe
Model3.add(MaxPooling2D(pool_size=(2,2)))

Model3.add(Conv2D(filters=32,kernel_size=(4,4),padding='same',data_format='channe
Model3.add(Conv2D(filters=32,kernel_size=(2,2),padding='same',data_format='channe
Model3.add(Conv2D(filters=16,kernel_size=(2,2),padding='same',data_format='channe
Model3.add(BatchNormalization(axis=1))
Model3.add(Dropout(0.5))

Model3.add(Flatten())
Model3.add(Dense(60,activation='relu'))
Model3.add(Dense(30,activation='relu'))
Model3.add(Dense(10,activation='softmax'))

Model3.compile(loss='categorical_crossentropy', optimizer='adam',metrics=['accura

Model3.fit(x3d_train,y_train, validation_data=(x3d_test,y_test),epochs = 30,batch_
```

```
Train on 50000 samples, validate on 10000 samples
Epoch 1/30
50000/50000 [==============================] - 1634s 33ms/step - loss: 1.6541 -
acc: 0.3958 - val_loss: 3.6330 - val_acc: 0.3291
Epoch 2/30
50000/50000 [==============================] - 1624s 32ms/step - loss: 1.3479 -
acc: 0.5103 - val_loss: 1.3301 - val_acc: 0.5162
Epoch 3/30
50000/50000 [==============================] - 1613s 32ms/step - loss: 1.1849 -
acc: 0.5745 - val_loss: 1.2925 - val_acc: 0.5363
Epoch 4/30
50000/50000 [==============================] - 1618s 32ms/step - loss: 1.0619 -
acc: 0.6226 - val_loss: 1.0897 - val_acc: 0.6157
Epoch 5/30
50000/50000 [==============================] - 1618s 32ms/step - loss: 0.9673 -
acc: 0.6572 - val_loss: 1.1977 - val_acc: 0.5849
Epoch 6/30
50000/50000 [==============================] - 1607s 32ms/step - loss: 0.8937 -
acc: 0.6836 - val_loss: 0.9780 - val_acc: 0.6571
Epoch 7/30
50000/50000 [==============================] - 1612s 32ms/step - loss: 0.8355 -
acc: 0.7046 - val_loss: 0.9990 - val_acc: 0.6509
Epoch 8/30
50000/50000 [==============================] - 1717s 34ms/step - loss: 0.7818 -
acc: 0.7253 - val_loss: 1.2301 - val_acc: 0.5933
Epoch 9/30
50000/50000 [==============================] - 1716s 34ms/step - loss: 0.7419 -
acc: 0.7370 - val_loss: 0.9917 - val_acc: 0.6647
Epoch 10/30
50000/50000 [==============================] - 1732s 35ms/step - loss: 0.6985 -
acc: 0.7536 - val_loss: 0.8762 - val_acc: 0.7010
Epoch 11/30
50000/50000 [==============================] - 1745s 35ms/step - loss: 0.6688 -
```

```
acc: 0.7630 - val_loss: 0.8428 - val_acc: 0.7144
Epoch 12/30
50000/50000 [==============================] - 1795s 36ms/step - loss: 0.6377 -
acc: 0.7751 - val_loss: 0.8989 - val_acc: 0.6985
Epoch 13/30
50000/50000 [==============================] - 1741s 35ms/step - loss: 0.6089 -
acc: 0.7855 - val_loss: 0.8314 - val_acc: 0.7143
Epoch 14/30
50000/50000 [==============================] - 1736s 35ms/step - loss: 0.5874 -
acc: 0.7934 - val_loss: 0.8765 - val_acc: 0.7082
Epoch 15/30
50000/50000 [==============================] - 1747s 35ms/step - loss: 0.5707 -
acc: 0.7985 - val_loss: 0.9762 - val_acc: 0.6858
Epoch 16/30
50000/50000 [==============================] - 1755s 35ms/step - loss: 0.5484 -
acc: 0.8066 - val_loss: 0.8010 - val_acc: 0.7324
Epoch 17/30
50000/50000 [==============================] - 1714s 34ms/step - loss: 0.5197 -
acc: 0.8155 - val_loss: 1.6506 - val_acc: 0.6557
Epoch 18/30
50000/50000 [==============================] - 1746s 35ms/step - loss: 0.5157 -
acc: 0.8178 - val_loss: 0.9260 - val_acc: 0.7015
Epoch 19/30
50000/50000 [==============================] - 1743s 35ms/step - loss: 0.4871 -
acc: 0.8286 - val_loss: 0.8662 - val_acc: 0.7294
Epoch 20/30
50000/50000 [==============================] - 1740s 35ms/step - loss: 0.4703 -
acc: 0.8344 - val_loss: 0.7881 - val_acc: 0.7416
Epoch 21/30
50000/50000 [==============================] - 1740s 35ms/step - loss: 0.4628 -
acc: 0.8357 - val_loss: 0.8728 - val_acc: 0.7270
Epoch 22/30
50000/50000 [==============================] - 1724s 34ms/step - loss: 0.4443 -
acc: 0.8414 - val_loss: 0.8528 - val_acc: 0.7329
Epoch 23/30
50000/50000 [==============================] - 1611s 32ms/step - loss: 0.4321 -
acc: 0.8483 - val_loss: 0.9181 - val_acc: 0.7144
Epoch 24/30
50000/50000 [==============================] - 2055s 41ms/step - loss: 0.4311 -
acc: 0.8470 - val_loss: 0.8615 - val_acc: 0.7288
Epoch 25/30
50000/50000 [==============================] - 1767s 35ms/step - loss: 0.4050 -
acc: 0.8549 - val_loss: 0.9034 - val_acc: 0.7266
Epoch 26/30
50000/50000 [==============================] - 1737s 35ms/step - loss: 0.3953 -
acc: 0.8593 - val_loss: 0.8346 - val_acc: 0.7486
Epoch 27/30
50000/50000 [==============================] - 1733s 35ms/step - loss: 0.3840 -
acc: 0.8632 - val_loss: 0.8770 - val_acc: 0.7281
Epoch 28/30
50000/50000 [==============================] - 1730s 35ms/step - loss: 0.3708 -
acc: 0.8685 - val_loss: 0.8825 - val_acc: 0.7414
Epoch 29/30
50000/50000 [==============================] - 1736s 35ms/step - loss: 0.3661 -
acc: 0.8707 - val_loss: 0.8783 - val_acc: 0.7263
Epoch 30/30
```

```
50000/50000 [==============================] - 1742s 35ms/step - loss: 0.3560 -
acc: 0.8733 - val_loss: 0.9214 - val_acc: 0.7308
```

Out[54]: <keras.callbacks.History at 0x2730e7234a8>

```
Architecture:
Conv2D 128 filter(8,8) --> BatchNormalize --> Conv2D 64 filter(4,4) -->
MaxPooling(2,2) --> Conv2D 32 filter(4,4) --> Conv2D 32 filter(2,2) --> Conv2D
16 filter(2,2) --> BatchNormalize --> Dropout(0.5) --> Flatten --> Dense 60 -->
Dense 30 --> Dense 10
Accuracy = 0.7308 = 73.08%


Result
Train on 50000 samples, validate on 10000 samples
Epoch 1/30
50000/50000 [==============================] - 1634s 33ms/step - loss: 1.6541 -
acc: 0.3958 - val_loss: 3.6330 - val_acc: 0.3291
Epoch 2/30
50000/50000 [==============================] - 1624s 32ms/step - loss: 1.3479 -
acc: 0.5103 - val_loss: 1.3301 - val_acc: 0.5162
Epoch 3/30
50000/50000 [==============================] - 1613s 32ms/step - loss: 1.1849 -
acc: 0.5745 - val_loss: 1.2925 - val_acc: 0.5363
Epoch 4/30
50000/50000 [==============================] - 1618s 32ms/step - loss: 1.0619 -
acc: 0.6226 - val_loss: 1.0897 - val_acc: 0.6157
Epoch 5/30
50000/50000 [==============================] - 1618s 32ms/step - loss: 0.9673 -
acc: 0.6572 - val_loss: 1.1977 - val_acc: 0.5849
Epoch 6/30
50000/50000 [==============================] - 1607s 32ms/step - loss: 0.8937 -
acc: 0.6836 - val_loss: 0.9780 - val_acc: 0.6571
Epoch 7/30
50000/50000 [==============================] - 1612s 32ms/step - loss: 0.8355 -
acc: 0.7046 - val_loss: 0.9990 - val_acc: 0.6509
Epoch 8/30
50000/50000 [==============================] - 1717s 34ms/step - loss: 0.7818 -
acc: 0.7253 - val_loss: 1.2301 - val_acc: 0.5933
Epoch 9/30
50000/50000 [==============================] - 1716s 34ms/step - loss: 0.7419 -
acc: 0.7370 - val_loss: 0.9917 - val_acc: 0.6647
Epoch 10/30
50000/50000 [==============================] - 1732s 35ms/step - loss: 0.6985 -
acc: 0.7536 - val_loss: 0.8762 - val_acc: 0.7010
Epoch 11/30
50000/50000 [==============================] - 1745s 35ms/step - loss: 0.6688 -
acc: 0.7630 - val_loss: 0.8428 - val_acc: 0.7144
Epoch 12/30
50000/50000 [==============================] - 1795s 36ms/step - loss: 0.6377 -
acc: 0.7751 - val_loss: 0.8989 - val_acc: 0.6985
Epoch 13/30
50000/50000 [==============================] - 1741s 35ms/step - loss: 0.6089 -
acc: 0.7855 - val_loss: 0.8314 - val_acc: 0.7143
Epoch 14/30
50000/50000 [==============================] - 1736s 35ms/step - loss: 0.5874 -
acc: 0.7934 - val_loss: 0.8765 - val_acc: 0.7082
```

```
Epoch 15/30
50000/50000 [==============================] - 1747s 35ms/step - loss: 0.5707 -
acc: 0.7985 - val_loss: 0.9762 - val_acc: 0.6858
Epoch 16/30
50000/50000 [==============================] - 1755s 35ms/step - loss: 0.5484 -
acc: 0.8066 - val_loss: 0.8010 - val_acc: 0.7324
Epoch 17/30
50000/50000 [==============================] - 1714s 34ms/step - loss: 0.5197 -
acc: 0.8155 - val_loss: 1.6506 - val_acc: 0.6557
Epoch 18/30
50000/50000 [==============================] - 1746s 35ms/step - loss: 0.5157 -
acc: 0.8178 - val_loss: 0.9260 - val_acc: 0.7015
Epoch 19/30
50000/50000 [==============================] - 1743s 35ms/step - loss: 0.4871 -
acc: 0.8286 - val_loss: 0.8662 - val_acc: 0.7294
Epoch 20/30
50000/50000 [==============================] - 1740s 35ms/step - loss: 0.4703 -
acc: 0.8344 - val_loss: 0.7881 - val_acc: 0.7416
Epoch 21/30
50000/50000 [==============================] - 1740s 35ms/step - loss: 0.4628 -
acc: 0.8357 - val_loss: 0.8728 - val_acc: 0.7270
Epoch 22/30
50000/50000 [==============================] - 1724s 34ms/step - loss: 0.4443 -
acc: 0.8414 - val_loss: 0.8528 - val_acc: 0.7329
Epoch 23/30
50000/50000 [==============================] - 1611s 32ms/step - loss: 0.4321 -
acc: 0.8483 - val_loss: 0.9181 - val_acc: 0.7144
Epoch 24/30
50000/50000 [==============================] - 2055s 41ms/step - loss: 0.4311 -
acc: 0.8470 - val_loss: 0.8615 - val_acc: 0.7288
Epoch 25/30
50000/50000 [==============================] - 1767s 35ms/step - loss: 0.4050 -
acc: 0.8549 - val_loss: 0.9034 - val_acc: 0.7266
Epoch 26/30
50000/50000 [==============================] - 1737s 35ms/step - loss: 0.3953 -
acc: 0.8593 - val_loss: 0.8346 - val_acc: 0.7486
Epoch 27/30
50000/50000 [==============================] - 1733s 35ms/step - loss: 0.3840 -
acc: 0.8632 - val_loss: 0.8770 - val_acc: 0.7281
Epoch 28/30
50000/50000 [==============================] - 1730s 35ms/step - loss: 0.3708 -
acc: 0.8685 - val_loss: 0.8825 - val_acc: 0.7414
Epoch 29/30
50000/50000 [==============================] - 1736s 35ms/step - loss: 0.3661 -
acc: 0.8707 - val_loss: 0.8783 - val_acc: 0.7263
Epoch 30/30
50000/50000 [==============================] - 1742s 35ms/step - loss: 0.3560 -
acc: 0.8733 - val_loss: 0.9214 - val_acc: 0.7308
```

จากผลลัพธ์ พบว่า model มี overfit ลดลง แต่ ไม่สามารถทำให้ accuracy สูงเกินกว่า 75% ได้ จึงคิดว่า จะเปลี่ยนตำแหน่งของ BatchNormalize ตัวแรกเพราะคิดว่าเร็วเกินไปที่จะ Normalize ข้อมูล และเพิ่ม Convolution layer เพิ่มอีก 1 เพื่อให้ model มี accuracy สูงขึ้นได้ และลด Dropout ลงเพราะคิดว่ามีผลทำให้ model ได้ทิ้ง feature ที่สำคัญไปทำให้ accuracy ไมาสามารถเพิ่มขึ้นได้ จากแนวคิดทั้งหมดจึงได้ model2 ออกมา

Model2 (FINAL MODEL) ACCURACY = 75.17%

In [62]:
```
Model2 = Sequential()

Model2.add(Conv2D(filters=128,kernel_size=(8,8),padding='same',data_format='chann
Model2.add(Conv2D(filters=64,kernel_size=(4,4),padding='same',data_format='channe
Model2.add(BatchNormalization(axis=1))
Model2.add(MaxPooling2D(pool_size=(2,2)))

Model2.add(Conv2D(filters=64,kernel_size=(4,4),padding='same',data_format='channe
Model2.add(Conv2D(filters=32,kernel_size=(4,4),padding='same',data_format='channe
Model2.add(Conv2D(filters=16,kernel_size=(2,2),padding='same',data_format='channe
Model2.add(MaxPooling2D(pool_size=(2,2)))
Model2.add(BatchNormalization(axis=1))
Model2.add(Dropout(0.2))

Model2.add(Flatten())
Model2.add(Dense(20,activation='relu'))
Model2.add(Dense(10,activation='softmax'))

Model2.compile(loss='categorical_crossentropy', optimizer='adam',metrics=['accura

hist2 = Model2.fit(x3d_train,y_train, validation_data=(x3d_test,y_test),epochs =
score2 = Model2.evaluate(x3d_test,y_test,batch_size = 32)
```

```
Train on 50000 samples, validate on 10000 samples
Epoch 1/30
50000/50000 [==============================] - 1500s 30ms/step - loss: 1.7190 -
acc: 0.3618 - val_loss: 1.7087 - val_acc: 0.4184
Epoch 2/30
50000/50000 [==============================] - 1491s 30ms/step - loss: 1.3754 -
acc: 0.4966 - val_loss: 1.6545 - val_acc: 0.4204
Epoch 3/30
50000/50000 [==============================] - 1483s 30ms/step - loss: 1.1870 -
acc: 0.5737 - val_loss: 1.4854 - val_acc: 0.4950
Epoch 4/30
50000/50000 [==============================] - 1482s 30ms/step - loss: 1.0339 -
acc: 0.6314 - val_loss: 1.0369 - val_acc: 0.6353
Epoch 5/30
50000/50000 [==============================] - 1505s 30ms/step - loss: 0.9272 -
acc: 0.6697 - val_loss: 0.9406 - val_acc: 0.6698
Epoch 6/30
50000/50000 [==============================] - 1483s 30ms/step - loss: 0.8512 -
acc: 0.6995 - val_loss: 1.0637 - val_acc: 0.6409
Epoch 7/30
50000/50000 [==============================] - 1482s 30ms/step - loss: 0.7946 -
acc: 0.7186 - val_loss: 0.8451 - val_acc: 0.7028
Epoch 8/30
50000/50000 [==============================] - 1481s 30ms/step - loss: 0.7479 -
acc: 0.7364 - val_loss: 0.8510 - val_acc: 0.7058
Epoch 9/30
50000/50000 [==============================] - 1482s 30ms/step - loss: 0.7065 -
acc: 0.7489 - val_loss: 0.8792 - val_acc: 0.6932
Epoch 10/30
50000/50000 [==============================] - 1483s 30ms/step - loss: 0.6760 -
acc: 0.7609 - val_loss: 0.8598 - val_acc: 0.7058
Epoch 11/30
```

```
50000/50000 [==============================] - 1483s 30ms/step - loss: 0.6442 -
acc: 0.7711 - val_loss: 0.8485 - val_acc: 0.7139
Epoch 12/30
50000/50000 [==============================] - 1483s 30ms/step - loss: 0.6109 -
acc: 0.7830 - val_loss: 0.9642 - val_acc: 0.6800
Epoch 13/30
50000/50000 [==============================] - 1483s 30ms/step - loss: 0.5864 -
acc: 0.7910 - val_loss: 0.9193 - val_acc: 0.6991
Epoch 14/30
50000/50000 [==============================] - 1438s 29ms/step - loss: 0.5628 -
acc: 0.8007 - val_loss: 0.8046 - val_acc: 0.7355
Epoch 15/30
50000/50000 [==============================] - 1377s 28ms/step - loss: 0.5386 -
acc: 0.8088 - val_loss: 0.8892 - val_acc: 0.7091
Epoch 16/30
50000/50000 [==============================] - 1381s 28ms/step - loss: 0.5174 -
acc: 0.8160 - val_loss: 0.8525 - val_acc: 0.7202
Epoch 17/30
50000/50000 [==============================] - 1379s 28ms/step - loss: 0.4974 -
acc: 0.8218 - val_loss: 0.7958 - val_acc: 0.7427
Epoch 18/30
50000/50000 [==============================] - 1380s 28ms/step - loss: 0.4819 -
acc: 0.8287 - val_loss: 0.7817 - val_acc: 0.7417
Epoch 19/30
50000/50000 [==============================] - 1385s 28ms/step - loss: 0.4561 -
acc: 0.8376 - val_loss: 0.8169 - val_acc: 0.7413
Epoch 20/30
50000/50000 [==============================] - 1384s 28ms/step - loss: 0.4414 -
acc: 0.8419 - val_loss: 0.8369 - val_acc: 0.7443
Epoch 21/30
50000/50000 [==============================] - 1377s 28ms/step - loss: 0.4240 -
acc: 0.8489 - val_loss: 0.8434 - val_acc: 0.7311
Epoch 22/30
50000/50000 [==============================] - 1376s 28ms/step - loss: 0.4102 -
acc: 0.8550 - val_loss: 0.8658 - val_acc: 0.7334
Epoch 23/30
50000/50000 [==============================] - 1430s 29ms/step - loss: 0.3940 -
acc: 0.8612 - val_loss: 0.8969 - val_acc: 0.7357
Epoch 24/30
50000/50000 [==============================] - 1403s 28ms/step - loss: 0.3749 -
acc: 0.8650 - val_loss: 0.8846 - val_acc: 0.7356
Epoch 25/30
50000/50000 [==============================] - 1385s 28ms/step - loss: 0.3688 -
acc: 0.8669 - val_loss: 1.0343 - val_acc: 0.7155
Epoch 26/30
50000/50000 [==============================] - 1382s 28ms/step - loss: 0.3570 -
acc: 0.8718 - val_loss: 0.8489 - val_acc: 0.7492
Epoch 27/30
50000/50000 [==============================] - 1384s 28ms/step - loss: 0.3401 -
acc: 0.8774 - val_loss: 0.8902 - val_acc: 0.7419
Epoch 28/30
50000/50000 [==============================] - 1380s 28ms/step - loss: 0.3309 -
acc: 0.8807 - val_loss: 0.9544 - val_acc: 0.7421
Epoch 29/30
50000/50000 [==============================] - 1389s 28ms/step - loss: 0.3200 -
acc: 0.8842 - val_loss: 0.9410 - val_acc: 0.7390
Epoch 30/30
```

```
50000/50000 [==============================] - 1393s 28ms/step - loss: 0.3105 -
acc: 0.8879 - val_loss: 0.8948 - val_acc: 0.7517
10000/10000 [==============================] - 111s 11ms/step
```

```
Architecture:
conv2D 128 filter(8x8) --> conv2D 64 filter(4x4) --> conv2D 32 filter(4,4) -->
BatchNormalize --> Maxpooling2D(2,2) --> conv2D 64 filter(4x4) --> conv2D 32
filter(4x4) --> conv2D 16 filter(4x4) --> Maxpooling(2,2) --> Dropout(0.2) -->
Flatten --> Dense(20)-->Dense(10)
Accuracy = 0.7517 = 75.17%


Result
Train on 50000 samples, validate on 10000 samples
Epoch 1/30
50000/50000 [==============================] - 1500s 30ms/step - loss: 1.7190 -
acc: 0.3618 - val_loss: 1.7087 - val_acc: 0.4184
Epoch 2/30
50000/50000 [==============================] - 1491s 30ms/step - loss: 1.3754 -
acc: 0.4966 - val_loss: 1.6545 - val_acc: 0.4204
Epoch 3/30
50000/50000 [==============================] - 1483s 30ms/step - loss: 1.1870 -
acc: 0.5737 - val_loss: 1.4854 - val_acc: 0.4950
Epoch 4/30
50000/50000 [==============================] - 1482s 30ms/step - loss: 1.0339 -
acc: 0.6314 - val_loss: 1.0369 - val_acc: 0.6353
Epoch 5/30
50000/50000 [==============================] - 1505s 30ms/step - loss: 0.9272 -
acc: 0.6697 - val_loss: 0.9406 - val_acc: 0.6698
Epoch 6/30
50000/50000 [==============================] - 1483s 30ms/step - loss: 0.8512 -
acc: 0.6995 - val_loss: 1.0637 - val_acc: 0.6409
Epoch 7/30
50000/50000 [==============================] - 1482s 30ms/step - loss: 0.7946 -
acc: 0.7186 - val_loss: 0.8451 - val_acc: 0.7028
Epoch 8/30
50000/50000 [==============================] - 1481s 30ms/step - loss: 0.7479 -
acc: 0.7364 - val_loss: 0.8510 - val_acc: 0.7058
Epoch 9/30
50000/50000 [==============================] - 1482s 30ms/step - loss: 0.7065 -
acc: 0.7489 - val_loss: 0.8792 - val_acc: 0.6932
Epoch 10/30
50000/50000 [==============================] - 1483s 30ms/step - loss: 0.6760 -
acc: 0.7609 - val_loss: 0.8598 - val_acc: 0.7058
Epoch 11/30
50000/50000 [==============================] - 1483s 30ms/step - loss: 0.6442 -
acc: 0.7711 - val_loss: 0.8485 - val_acc: 0.7139
Epoch 12/30
50000/50000 [==============================] - 1483s 30ms/step - loss: 0.6109 -
acc: 0.7830 - val_loss: 0.9642 - val_acc: 0.6800
Epoch 13/30
50000/50000 [==============================] - 1483s 30ms/step - loss: 0.5864 -
acc: 0.7910 - val_loss: 0.9193 - val_acc: 0.6991
Epoch 14/30
50000/50000 [==============================] - 1438s 29ms/step - loss: 0.5628 -
acc: 0.8007 - val_loss: 0.8046 - val_acc: 0.7355
Epoch 15/30
```

```
50000/50000 [==============================] - 1377s 28ms/step - loss: 0.5386 -
acc: 0.8088 - val_loss: 0.8892 - val_acc: 0.7091
Epoch 16/30
50000/50000 [==============================] - 1381s 28ms/step - loss: 0.5174 -
acc: 0.8160 - val_loss: 0.8525 - val_acc: 0.7202
Epoch 17/30
50000/50000 [==============================] - 1379s 28ms/step - loss: 0.4974 -
acc: 0.8218 - val_loss: 0.7958 - val_acc: 0.7427
Epoch 18/30
50000/50000 [==============================] - 1380s 28ms/step - loss: 0.4819 -
acc: 0.8287 - val_loss: 0.7817 - val_acc: 0.7417
Epoch 19/30
50000/50000 [==============================] - 1385s 28ms/step - loss: 0.4561 -
acc: 0.8376 - val_loss: 0.8169 - val_acc: 0.7413
Epoch 20/30
50000/50000 [==============================] - 1384s 28ms/step - loss: 0.4414 -
acc: 0.8419 - val_loss: 0.8369 - val_acc: 0.7443
Epoch 21/30
50000/50000 [==============================] - 1377s 28ms/step - loss: 0.4240 -
acc: 0.8489 - val_loss: 0.8434 - val_acc: 0.7311
Epoch 22/30
50000/50000 [==============================] - 1376s 28ms/step - loss: 0.4102 -
acc: 0.8550 - val_loss: 0.8658 - val_acc: 0.7334
Epoch 23/30
50000/50000 [==============================] - 1430s 29ms/step - loss: 0.3940 -
acc: 0.8612 - val_loss: 0.8969 - val_acc: 0.7357
Epoch 24/30
50000/50000 [==============================] - 1403s 28ms/step - loss: 0.3749 -
acc: 0.8650 - val_loss: 0.8846 - val_acc: 0.7356
Epoch 25/30
50000/50000 [==============================] - 1385s 28ms/step - loss: 0.3688 -
acc: 0.8669 - val_loss: 1.0343 - val_acc: 0.7155
Epoch 26/30
50000/50000 [==============================] - 1382s 28ms/step - loss: 0.3570 -
acc: 0.8718 - val_loss: 0.8489 - val_acc: 0.7492
Epoch 27/30
50000/50000 [==============================] - 1384s 28ms/step - loss: 0.3401 -
acc: 0.8774 - val_loss: 0.8902 - val_acc: 0.7419
Epoch 28/30
50000/50000 [==============================] - 1380s 28ms/step - loss: 0.3309 -
acc: 0.8807 - val_loss: 0.9544 - val_acc: 0.7421
Epoch 29/30
50000/50000 [==============================] - 1389s 28ms/step - loss: 0.3200 -
acc: 0.8842 - val_loss: 0.9410 - val_acc: 0.7390
Epoch 30/30
50000/50000 [==============================] - 1393s 28ms/step - loss: 0.3105 -
acc: 0.8879 - val_loss: 0.8948 - val_acc: 0.7517
```