In this notebook will check how other parameters will affect the accuracy of the model

1. kernel_initializer
2. kernal_regularizer

In [1]:
```
import pickle
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from keras.models import Sequential
from keras.layers import Conv2D
from keras.layers import MaxPooling2D
from keras.layers import Flatten, BatchNormalization
from keras.layers import Dense, Dropout
from keras import regularizers
from keras.optimizers import SGD
from keras.preprocessing.image import ImageDataGenerator
from keras.utils import np_utils
import keras
```

```
Using TensorFlow backend.
```

In [2]:
```
def load_train_data(n):
    with open('data_batch_'+ str(n), 'rb') as file:
        batch = pickle.load(file, encoding='latin1')

    features = batch['data']
    Target = batch['labels']
    return features, Target
```

In [3]:
```
batch_1, Target_1 = load_train_data(1)
```

```
batch_2, Target_2 = load_train_data(2)
batch_3, Target_3 = load_train_data(3)
batch_4, Target_4 = load_train_data(4)
batch_5, Target_5 = load_train_data(5)
```

In [4]:
```
with open('test_batch', 'rb') as file:
    batch = pickle.load(file, encoding='latin1')
X_test = batch['data']
y_test = batch['labels']
print('test batch data and label data shape are',X_test.shape,len(y_tes
t))
```

```
test batch data and label data shape are (10000, 3072) 10000
```

In [5]:
```
X_train = np.append(batch_1, batch_2,axis=0)
X_train = np.append(X_train, batch_3,axis=0)
X_train = np.append(X_train, batch_4,axis=0)
X_train = np.append(X_train, batch_5,axis=0)
y_train = np.append(Target_1, Target_2,axis=0)
y_train = np.append(y_train, Target_3,axis=0)
y_train = np.append(y_train, Target_4,axis=0)
y_train = np.append(y_train, Target_5,axis=0)
X_train = X_train.reshape((len(X_train), 3, 32, 32)).transpose(0,2,3,1)
y_train = np_utils.to_categorical(y_train, 10)
X_test = X_test.reshape((len(X_test), 3, 32, 32)).transpose(0,2,3,1)
y_test = np_utils.to_categorical(y_test, 10)
X_train = X_train.astype('float32')
X_test= X_test.astype('float32')
X_train= X_train / 255.0
X_test= X_test/ 255.0
```

Model 5

Let check how kernel_initializer and regularizer will affect the model with default parameters. In the previous model used'he_normal' and l2(0.001)

In [7]:
```
model5 = Sequential()
model5.add(Conv2D(64, (3, 3), activation='relu', input_shape=(32, 32, 3
)))
model5.add(Conv2D(64, (3, 3), activation='relu'))
model5.add(MaxPooling2D((2, 2)))
model5.add(Conv2D(64, (3, 3), activation='relu'))
model5.add(Conv2D(64, (3, 3), activation='relu'))
model5.add(MaxPooling2D((2, 2)))
model5.add(Conv2D(64, (3, 3), activation='relu'))
model5.add(MaxPooling2D((2, 2)))
model5.add(Flatten())
model5.add(Dense(128, activation='relu'))
model5.add(Dense(10, activation='softmax'))
model5.summary()
```

```
WARNING:tensorflow:From C:\Users\Dhanajayan\Anaconda3\lib\site-packages
\tensorflow\python\framework\op_def_library.py:263: colocate_with (from
tensorflow.python.framework.ops) is deprecated and will be removed in a
future version.
Instructions for updating:
Colocations handled automatically by placer.
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_1 (Conv2D)            (None, 30, 30, 64)        1792      
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 28, 28, 64)        36928     
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 14, 14, 64)        0         
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 12, 12, 64)        36928     
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 10, 10, 64)        36928     
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 5, 5, 64)          0         
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 3, 3, 64)          36928     
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 1, 1, 64)          0         
_________________________________________________________________
```

```
flatten_1 (Flatten)          (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               8320
_________________________________________________________________
dense_2 (Dense)              (None, 10)                1290
=================================================================
Total params: 159,114
Trainable params: 159,114
Non-trainable params: 0
_________________________________________________________________
```

In [8]:
```
epochs = 10
sgd = SGD(lr=1e-2, momentum=0.9, decay=1e-2/epochs)
model5.compile(optimizer=sgd, loss='categorical_crossentropy', metrics=
['accuracy'])
model5.fit(X_train,y_train,epochs=epochs,batch_size = 32)
```

```
WARNING:tensorflow:From C:\Users\Dhanajayan\Anaconda3\lib\site-packages
\tensorflow\python\ops\math_ops.py:3066: to_int32 (from tensorflow.pyth
on.ops.math_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.cast instead.
Epoch 1/10
50000/50000 [==============================] - 330s 7ms/step - loss: 1.
8360 - acc: 0.3230
Epoch 2/10
50000/50000 [==============================] - 317s 6ms/step - loss: 1.
4457 - acc: 0.4724
Epoch 3/10
50000/50000 [==============================] - 363s 7ms/step - loss: 1.
2767 - acc: 0.5431
Epoch 4/10
50000/50000 [==============================] - 251s 5ms/step - loss: 1.
1520 - acc: 0.5907
Epoch 5/10
50000/50000 [==============================] - 257s 5ms/step - loss: 1.
0623 - acc: 0.6258
Epoch 6/10
50000/50000 [==============================] - 260s 5ms/step - loss: 0.
9918 - acc: 0.6529
```

```
Epoch 7/10
50000/50000 [==============================] - 265s 5ms/step - loss: 0.
9399 - acc: 0.6695
Epoch 8/10
50000/50000 [==============================] - 268s 5ms/step - loss: 0.
8917 - acc: 0.6872
Epoch 9/10
50000/50000 [==============================] - 274s 5ms/step - loss: 0.
8513 - acc: 0.7029
Epoch 10/10
50000/50000 [==============================] - 310s 6ms/step - loss: 0.
8185 - acc: 0.7134
```

Out[8]:
```
<keras.callbacks.History at 0x250a54e3be0>
```

In [9]:
```
test_loss,test_acc = model5.evaluate(X_test,y_test)
test_acc
```

```
10000/10000 [==============================] - 27s 3ms/step
```

Out[9]:
```
0.6503
```

Observation

The difference in train and test accuracy is 4% By comparing model 1 and model 5 the default kernel_initializer and kernel_regularizer has 4% less train accuracy in model 5 and 2% less test accuracy in model 1 The differene in computation is not that much

Model 6

In [11]:
```
model6 = Sequential()
model6.add(Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3
)))
model6.add(Conv2D(32, (3, 3), activation='relu'))
```

```
model6.add(MaxPooling2D((2, 2)))
model6.add(Conv2D(64, (3, 3), activation='relu'))
model6.add(Conv2D(64, (3, 3), activation='relu'))
model6.add(MaxPooling2D((2, 2)))
model6.add(Conv2D(128, (3, 3), activation='relu'))
model6.add(MaxPooling2D((2, 2)))
model6.add(Flatten())
model6.add(Dense(128, activation='relu'))
model6.add(Dense(10, activation='softmax'))
model6.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_6 (Conv2D)            (None, 30, 30, 32)        896       
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 28, 28, 32)        9248      
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 14, 14, 32)        0         
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 12, 12, 64)        18496     
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 10, 10, 64)        36928     
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 5, 5, 64)          0         
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 3, 3, 128)         73856     
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 1, 1, 128)         0         
_________________________________________________________________
flatten_2 (Flatten)          (None, 128)               0         
_________________________________________________________________
dense_3 (Dense)              (None, 128)               16512     
_________________________________________________________________
dense_4 (Dense)              (None, 10)                1290      
=================================================================
Total params: 157,226
Trainable params: 157,226
Non-trainable params: 0
_________________________________________________________________
```

In [12]:
```
epochs = 10
sgd = SGD(lr=1e-2, momentum=0.9, decay=1e-2/epochs)
model6.compile(optimizer=sgd, loss='categorical_crossentropy', metrics=
['accuracy'])
model6.fit(X_train,y_train,epochs=epochs,batch_size = 32)
```

```
Epoch 1/10
50000/50000 [==============================] - 124s 2ms/step - loss: 1.
9033 - acc: 0.2951
Epoch 2/10
50000/50000 [==============================] - 125s 3ms/step - loss: 1.
4952 - acc: 0.4566
Epoch 3/10
50000/50000 [==============================] - 137s 3ms/step - loss: 1.
3413 - acc: 0.5148
Epoch 4/10
50000/50000 [==============================] - 128s 3ms/step - loss: 1.
2356 - acc: 0.5580
Epoch 5/10
50000/50000 [==============================] - 128s 3ms/step - loss: 1.
1542 - acc: 0.5890
Epoch 6/10
50000/50000 [==============================] - 128s 3ms/step - loss: 1.
0868 - acc: 0.6144
Epoch 7/10
50000/50000 [==============================] - 130s 3ms/step - loss: 1.
0295 - acc: 0.6365
Epoch 8/10
50000/50000 [==============================] - 132s 3ms/step - loss: 0.
9825 - acc: 0.6546
Epoch 9/10
50000/50000 [==============================] - 132s 3ms/step - loss: 0.
9402 - acc: 0.6689
Epoch 10/10
50000/50000 [==============================] - 132s 3ms/step - loss: 0.
9006 - acc: 0.6851
```

Out[12]:
```
<keras.callbacks.History at 0x250a54e3978>
```

```
In [13]: test_loss,test_acc = model6.evaluate(X_test,y_test)
         test_acc

         10000/10000 [==============================] - 11s 1ms/step

Out[13]: 0.6234

In [ ]:
```

Observation

- Comparing model 1, 2, 5 and 6 the model 5 and 6 are with default parameters gives less performance so we reject model 5 and 6 and consider model 1 and 2 for further analysis

```
In [ ]:
```