Model performance on batch size

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


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


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)


with open('test_batch', 'rb') as file:
    batch = pickle.load(file, encoding='latin1')
X_test = batch['data']
y_test = batch['labels']


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
```

```
Using TensorFlow backend.
```

Model 17

batch size 64

In [3]:
```
model17 = Sequential()
model17.add(Conv2D(64, (3, 3), activation='relu',kernel_initializer='he_normal',padding = 'same', in
put_shape=(32, 32, 3)))
model17.add(Conv2D(64, (3, 3), activation='relu',kernel_initializer='he_normal',padding = 'same'))
model17.add(MaxPooling2D((2, 2)))
model17.add(Conv2D(64, (3, 3), activation='relu',kernel_initializer='he_normal',padding = 'same'))
model17.add(Conv2D(64, (3, 3), activation='relu',kernel_initializer='he_normal',padding = 'same'))
model17.add(MaxPooling2D((2, 2)))
model17.add(Conv2D(64, (3, 3), activation='relu',kernel_initializer='he_normal',padding = 'same'))
model17.add(MaxPooling2D((2, 2)))
model17.add(Conv2D(64, (3, 3), activation='relu',kernel_initializer='he_normal',padding = 'same'))
model17.add(MaxPooling2D((2, 2)))
model17.add(Conv2D(64, (3, 3), activation='relu',kernel_initializer='he_normal',padding = 'same'))
model17.add(MaxPooling2D((2, 2)))
model17.add(Flatten())
model17.add(Dense(128, activation='relu'))
model17.add(Dropout(rate = 0.7))
model17.add(Dense(10, activation='softmax'))
model17.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_8 (Conv2D)            (None, 32, 32, 64)        1792
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 32, 32, 64)        36928
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 16, 16, 64)        0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 16, 16, 64)        36928
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 16, 16, 64)        36928
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 8, 8, 64)          0
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 8, 8, 64)          36928
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 4, 4, 64)          0
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 4, 4, 64)          36928
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 2, 2, 64)          0
_________________________________________________________________
conv2d_14 (Conv2D)           (None, 2, 2, 64)          36928
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 1, 1, 64)          0
_________________________________________________________________
flatten_2 (Flatten)          (None, 64)                0
_________________________________________________________________
dense_3 (Dense)              (None, 128)               8320
_________________________________________________________________
dropout_2 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_4 (Dense)              (None, 10)                1290
=================================================================
Total params: 232,970
Trainable params: 232,970
Non-trainable params: 0
_________________________________________________________________
```

In [4]:
```
epochs = 10
model17.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
model17.fit(X_train,y_train,epochs=epochs,batch_size = 64)
```

```
WARNING:tensorflow:From C:\Users\Dhanajayan\Anaconda3\lib\site-packages\tensorflow\python\ops\ma
th_ops.py:3066: to_int32 (from tensorflow.python.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 [==============================] - 263s 5ms/step - loss: 1.7821 - acc: 0.3367
Epoch 2/10
50000/50000 [==============================] - 293s 6ms/step - loss: 1.2750 - acc: 0.5495
Epoch 3/10
50000/50000 [==============================] - 297s 6ms/step - loss: 1.0559 - acc: 0.6357
Epoch 4/10
50000/50000 [==============================] - 326s 7ms/step - loss: 0.9251 - acc: 0.6856
Epoch 5/10
50000/50000 [==============================] - 319s 6ms/step - loss: 0.8322 - acc: 0.7212
Epoch 6/10
50000/50000 [==============================] - 318s 6ms/step - loss: 0.7426 - acc: 0.7516
Epoch 7/10
50000/50000 [==============================] - 330s 7ms/step - loss: 0.6818 - acc: 0.7748
Epoch 8/10
50000/50000 [==============================] - 385s 8ms/step - loss: 0.6257 - acc: 0.7941
Epoch 9/10
50000/50000 [==============================] - 368s 7ms/step - loss: 0.5747 - acc: 0.8115
Epoch 10/10
50000/50000 [==============================] - 378s 8ms/step - loss: 0.5316 - acc: 0.8266
```

Out[4]:
```
<keras.callbacks.History at 0x2641302c8d0>
```

In [5]:
```
test_loss,test_acc = model17.evaluate(X_test,y_test)
test_acc
```

```
10000/10000 [==============================] - 39s 4ms/step
```

Out[5]:
```
0.7454
```

Observation

The batch size 64 overfits the model than the model with batch size 32

Model 18

batch size - 128

In [7]:
```
model18 = Sequential()
model18.add(Conv2D(64, (3, 3), activation='relu',kernel_initializer='he_normal',padding = 'same', in
put_shape=(32, 32, 3)))
model18.add(Conv2D(64, (3, 3), activation='relu',kernel_initializer='he_normal',padding = 'same'))
model18.add(MaxPooling2D((2, 2)))
model18.add(Conv2D(64, (3, 3), activation='relu',kernel_initializer='he_normal',padding = 'same'))
model18.add(Conv2D(64, (3, 3), activation='relu',kernel_initializer='he_normal',padding = 'same'))
model18.add(MaxPooling2D((2, 2)))
model18.add(Conv2D(64, (3, 3), activation='relu',kernel_initializer='he_normal',padding = 'same'))
model18.add(MaxPooling2D((2, 2)))
model18.add(Conv2D(64, (3, 3), activation='relu',kernel_initializer='he_normal',padding = 'same'))
model18.add(MaxPooling2D((2, 2)))
model18.add(Conv2D(64, (3, 3), activation='relu',kernel_initializer='he_normal',padding = 'same'))
model18.add(MaxPooling2D((2, 2)))
model18.add(Flatten())
model18.add(Dense(128, activation='relu'))
model18.add(Dropout(rate = 0.7))
model18.add(Dense(10, activation='softmax'))
model18.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_15 (Conv2D)           (None, 32, 32, 64)        1792
_________________________________________________________________
conv2d_16 (Conv2D)           (None, 32, 32, 64)        36928
_________________________________________________________________
max_pooling2d_11 (MaxPooling (None, 16, 16, 64)        0
_________________________________________________________________
conv2d_17 (Conv2D)           (None, 16, 16, 64)        36928
_________________________________________________________________
conv2d_18 (Conv2D)           (None, 16, 16, 64)        36928
_________________________________________________________________
max_pooling2d_12 (MaxPooling (None, 8, 8, 64)          0
_________________________________________________________________
conv2d_19 (Conv2D)           (None, 8, 8, 64)          36928
_________________________________________________________________
max_pooling2d_13 (MaxPooling (None, 4, 4, 64)          0
_________________________________________________________________
conv2d_20 (Conv2D)           (None, 4, 4, 64)          36928
_________________________________________________________________
max_pooling2d_14 (MaxPooling (None, 2, 2, 64)          0
_________________________________________________________________
conv2d_21 (Conv2D)           (None, 2, 2, 64)          36928
_________________________________________________________________
max_pooling2d_15 (MaxPooling (None, 1, 1, 64)          0
_________________________________________________________________
flatten_3 (Flatten)          (None, 64)                0
_________________________________________________________________
dense_5 (Dense)              (None, 128)               8320
_________________________________________________________________
dropout_3 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_6 (Dense)              (None, 10)                1290
=================================================================
Total params: 232,970
Trainable params: 232,970
Non-trainable params: 0
_________________________________________________________________
```

In [8]:
```
epochs = 10
model18.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
model18.fit(X_train,y_train,epochs=epochs,batch_size = 128)
```

```
Epoch 1/10
50000/50000 [==============================] - 367s 7ms/step - loss: 1.8870 - acc: 0.2947
Epoch 2/10
50000/50000 [==============================] - 356s 7ms/step - loss: 1.4245 - acc: 0.4837
Epoch 3/10
50000/50000 [==============================] - 362s 7ms/step - loss: 1.2023 - acc: 0.5776
Epoch 4/10
50000/50000 [==============================] - 491s 10ms/step - loss: 1.0389 - acc: 0.6401
Epoch 5/10
50000/50000 [==============================] - 529s 11ms/step - loss: 0.9129 - acc: 0.6872
Epoch 6/10
50000/50000 [==============================] - 524s 10ms/step - loss: 0.8227 - acc: 0.7234
Epoch 7/10
50000/50000 [==============================] - 519s 10ms/step - loss: 0.7442 - acc: 0.7533
Epoch 8/10
50000/50000 [==============================] - 511s 10ms/step - loss: 0.6696 - acc: 0.7782
Epoch 9/10
50000/50000 [==============================] - 511s 10ms/step - loss: 0.6168 - acc: 0.7947
Epoch 10/10
50000/50000 [==============================] - 500s 10ms/step - loss: 0.5648 - acc: 0.8132
```

Out[8]:
```
<keras.callbacks.History at 0x26457abaeb8>
```

In [10]:
```
test_loss,test_acc = model18.evaluate(X_test,y_test)
test_acc
```

```
10000/10000 [==============================] - 49s 5ms/step
```

Out[10]:
```
0.7409
```

Observation

The batch size 64 and 128 almost works the same