Model performance with padding

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 9

- lets check the performance of the model with padding = 'same'

In [2]:
```
model9 = Sequential()
model9.add(Conv2D(64, (3, 3), activation='relu',kernel_initializer='he_normal',kernel_regularizer=re
gularizers.l2(0.001),padding = 'same', input_shape=(32, 32, 3)))
model9.add(Conv2D(64, (3, 3), activation='relu',kernel_initializer='he_normal',kernel_regularizer=re
gularizers.l2(0.001),padding = 'same'))
model9.add(MaxPooling2D((2, 2)))
model9.add(Conv2D(64, (3, 3), activation='relu',kernel_initializer='he_normal',kernel_regularizer=re
gularizers.l2(0.001),padding = 'same'))
model9.add(Conv2D(64, (3, 3), activation='relu',kernel_initializer='he_normal',kernel_regularizer=re
gularizers.l2(0.001),padding = 'same'))
model9.add(MaxPooling2D((2, 2)))
model9.add(Conv2D(64, (3, 3), activation='relu',kernel_initializer='he_normal',kernel_regularizer=re
gularizers.l2(0.001),padding = 'same'))
model9.add(MaxPooling2D((2, 2)))
model9.add(Flatten())
model9.add(Dense(128, activation='relu'))
model9.add(Dense(10, activation='softmax'))
model9.summary()
```

```
WARNING:tensorflow:From C:\Users\Dhanajayan\Anaconda3\lib\site-packages\tensorflow\python\framew
ork\op_def_library.py:263: colocate_with (from tensorflow.python.framework.ops) is deprecated an
d will be removed in a future version.
Instructions for updating:
Colocations handled automatically by placer.
```

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 32, 32, 64)	1792
conv2d_2 (Conv2D)	(None, 32, 32, 64)	36928
max_pooling2d_1 (MaxPooling2	(None, 16, 16, 64)	0
conv2d_3 (Conv2D)	(None, 16, 16, 64)	36928
conv2d_4 (Conv2D)	(None, 16, 16, 64)	36928
max_pooling2d_2 (MaxPooling2	(None, 8, 8, 64)	0
conv2d_5 (Conv2D)	(None, 8, 8, 64)	36928
max_pooling2d_3 (MaxPooling2	(None, 4, 4, 64)	0
flatten_1 (Flatten)	(None, 1024)	0
dense_1 (Dense)	(None, 128)	131200
dense_2 (Dense)	(None, 10)	1290

```
Total params: 281,994
Trainable params: 281,994
Non-trainable params: 0
```

In [3]:
```
epochs = 10
sgd = SGD(lr=1e-2, momentum=0.9, decay=1e-2/epochs)
model9.compile(optimizer=sgd, loss='categorical_crossentropy', metrics=['accuracy'])
model9.fit(X_train,y_train,epochs=epochs,batch_size = 32)
```

```
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 [==============================] - 422s 8ms/step - loss: 2.0384 - acc: 0.4528
Epoch 2/10
50000/50000 [==============================] - 395s 8ms/step - loss: 1.4851 - acc: 0.6282
Epoch 3/10
50000/50000 [==============================] - 368s 7ms/step - loss: 1.2553 - acc: 0.6990
Epoch 4/10
50000/50000 [==============================] - 407s 8ms/step - loss: 1.1193 - acc: 0.7405
Epoch 5/10
50000/50000 [==============================] - 344s 7ms/step - loss: 1.0284 - acc: 0.7650
Epoch 6/10
50000/50000 [==============================] - 351s 7ms/step - loss: 0.9592 - acc: 0.7822
Epoch 7/10
50000/50000 [==============================] - 352s 7ms/step - loss: 0.9070 - acc: 0.7973
Epoch 8/10
50000/50000 [==============================] - 346s 7ms/step - loss: 0.8611 - acc: 0.8110
Epoch 9/10
50000/50000 [==============================] - 343s 7ms/step - loss: 0.8220 - acc: 0.8226
Epoch 10/10
50000/50000 [==============================] - 345s 7ms/step - loss: 0.7903 - acc: 0.8313
```

Out[3]:
```
<keras.callbacks.History at 0x278eeabd470>
```

In [4]:
```
test_loss,test_acc = model9.evaluate(X_test,y_test)
test_acc
```

```
10000/10000 [==============================] - 25s 3ms/step
```

Out[4]:
```
0.7684
```

Observation

The padding 'same' parameter increased the efficieny comapared to default parameter

Model 10

In [7]:
```
model10= Sequential()
model10.add(Conv2D(32, (3, 3), activation='relu',kernel_initializer='he_normal',kernel_regularizer=r
egularizers.l2(0.001),padding = 'same', input_shape=(32, 32, 3)))
model10.add(Conv2D(32, (3, 3), activation='relu',kernel_initializer='he_normal',kernel_regularizer=r
egularizers.l2(0.001),padding = 'same'))
model10.add(MaxPooling2D((2, 2)))
model10.add(Conv2D(32, (3, 3), activation='relu',kernel_initializer='he_normal',kernel_regularizer=r
egularizers.l2(0.001),padding = 'same'))
model10.add(Conv2D(32, (3, 3), activation='relu',kernel_initializer='he_normal',kernel_regularizer=r
egularizers.l2(0.001),padding = 'same'))
model10.add(MaxPooling2D((2, 2)))
model10.add(Conv2D(32, (3, 3), activation='relu',kernel_initializer='he_normal',kernel_regularizer=r
egularizers.l2(0.001),padding = 'same'))
model10.add(MaxPooling2D((2, 2)))
model10.add(Flatten())
model10.add(Dense(64, activation='relu'))
model10.add(Dense(10, activation='softmax'))
model10.summary()
```

Layer (type)	Output Shape	Param #
conv2d_6 (Conv2D)	(None, 32, 32, 32)	896
conv2d_7 (Conv2D)	(None, 32, 32, 32)	9248
max_pooling2d_4 (MaxPooling2	(None, 16, 16, 32)	0
conv2d_8 (Conv2D)	(None, 16, 16, 32)	9248
conv2d_9 (Conv2D)	(None, 16, 16, 32)	9248
max_pooling2d_5 (MaxPooling2	(None, 8, 8, 32)	0
conv2d_10 (Conv2D)	(None, 8, 8, 32)	9248
max_pooling2d_6 (MaxPooling2	(None, 4, 4, 32)	0
flatten_2 (Flatten)	(None, 512)	0
dense_3 (Dense)	(None, 64)	32832
dense_4 (Dense)	(None, 10)	650

```
Total params: 71,370
Trainable params: 71,370
Non-trainable params: 0
```

In [8]:
```
epochs = 10
sgd = SGD(lr=1e-2, momentum=0.9, decay=1e-2/epochs)
model10.compile(optimizer=sgd, loss='categorical_crossentropy', metrics=['accuracy'])
model10.fit(X_train,y_train,epochs=epochs,batch_size = 32)
```

```
Epoch 1/10
50000/50000 [==============================] - 105s 2ms/step - loss: 1.8693 - acc: 0.4162
Epoch 2/10
50000/50000 [==============================] - 128s 3ms/step - loss: 1.3882 - acc: 0.5856
Epoch 3/10
50000/50000 [==============================] - 119s 2ms/step - loss: 1.2047 - acc: 0.6505
Epoch 4/10
50000/50000 [==============================] - 123s 2ms/step - loss: 1.1013 - acc: 0.6837
Epoch 5/10
50000/50000 [==============================] - 103s 2ms/step - loss: 1.0321 - acc: 0.7052
Epoch 6/10
50000/50000 [==============================] - 104s 2ms/step - loss: 0.9793 - acc: 0.7227
Epoch 7/10
50000/50000 [==============================] - 103s 2ms/step - loss: 0.9378 - acc: 0.7338
Epoch 8/10
50000/50000 [==============================] - 107s 2ms/step - loss: 0.9058 - acc: 0.7461
Epoch 9/10
50000/50000 [==============================] - 105s 2ms/step - loss: 0.8788 - acc: 0.7544
Epoch 10/10
50000/50000 [==============================] - 104s 2ms/step - loss: 0.8549 - acc: 0.7627
```

Out[8]:
```
<keras.callbacks.History at 0x27883ab76d8>
```

In [10]:
```
test_loss,test_acc = model10.evaluate(X_test,y_test)
test_acc
```

```
10000/10000 [==============================] - 10s 968us/step
```

Out[10]:
```
0.7309
```

Observation

The filter size 32 in model 10 gives less accuracy than the model 9. Therefore will proceed with the model9

In [ ]:
```

```