Model Performance with different optimizers

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 14

optimizer - rmsprop

In [3]:
```
model14 = Sequential()
model14.add(Conv2D(64, (3, 3), activation='relu',kernel_initializer='he_normal',padding = 'same', input_shape=(32, 32, 3)))
model14.add(Conv2D(64, (3, 3), activation='relu',kernel_initializer='he_normal',padding = 'same'))
model14.add(MaxPooling2D((2, 2)))
model14.add(Conv2D(64, (3, 3), activation='relu',kernel_initializer='he_normal',padding = 'same'))
model14.add(Conv2D(64, (3, 3), activation='relu',kernel_initializer='he_normal',padding = 'same'))
model14.add(MaxPooling2D((2, 2)))
model14.add(Conv2D(64, (3, 3), activation='relu',kernel_initializer='he_normal',padding = 'same'))
model14.add(Conv2D(64, (3, 3), activation='relu',kernel_initializer='he_normal',padding = 'same'))
model14.add(MaxPooling2D((2, 2)))
model14.add(Flatten())
model14.add(Dense(128, activation='relu'))
model14.add(Dropout(rate = 0.6))
model14.add(Dense(10, activation='softmax'))
model14.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
model14.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'])
model14.fit(X_train,y_train,epochs=epochs,batch_size = 32)
```

```
WARNING:tensorflow:From C:\Users\Dhana\ayan\Anaconda3\lib\site-packages\tensorflow\python\ops\math_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 [==============================] - 421s 8ms/step - loss: 1.6985 - acc: 0.3741
Epoch 2/10
50000/50000 [==============================] - 418s 8ms/step - loss: 1.1729 - acc: 0.5898
Epoch 3/10
50000/50000 [==============================] - 427s 9ms/step - loss: 0.9993 - acc: 0.6619
Epoch 4/10
50000/50000 [==============================] - 424s 8ms/step - loss: 0.9477 - acc: 0.6864
Epoch 5/10
50000/50000 [==============================] - 392s 8ms/step - loss: 0.9542 - acc: 0.6911
Epoch 6/10
50000/50000 [==============================] - 397s 8ms/step - loss: 1.0142 - acc: 0.6796
Epoch 7/10
50000/50000 [==============================] - 401s 8ms/step - loss: 1.1021 - acc: 0.6576
Epoch 8/10
50000/50000 [==============================] - 400s 8ms/step - loss: 1.1915 - acc: 0.6339
Epoch 9/10
50000/50000 [==============================] - 408s 8ms/step - loss: 1.2524 - acc: 0.6171
Epoch 10/10
50000/50000 [==============================] - 463s 9ms/step - loss: 1.3475 - acc: 0.5955
```

Out[4]:
```
<keras.callbacks.History at 0x2512e3f82b0>
```

In [5]:
```
test_loss,test_acc = model14.evaluate(X_test,y_test)
test_acc
```

```
10000/10000 [==============================] - 38s 4ms/step
```

Out[5]:
```
0.6292
```

Observation

In the above model the optimizer rmsprop not performed well as expected

Model 15

optimizer - adam

In [6]:
```
model15 = Sequential()
model15.add(Conv2D(64, (3, 3), activation='relu',kernel_initializer='he_normal',padding = 'same', input_shape=(32, 32, 3)))
model15.add(Conv2D(64, (3, 3), activation='relu',kernel_initializer='he_normal',padding = 'same'))
model15.add(MaxPooling2D((2, 2)))
model15.add(Conv2D(64, (3, 3), activation='relu',kernel_initializer='he_normal',padding = 'same'))
model15.add(Conv2D(64, (3, 3), activation='relu',kernel_initializer='he_normal',padding = 'same'))
model15.add(MaxPooling2D((2, 2)))
model15.add(Conv2D(64, (3, 3), activation='relu',kernel_initializer='he_normal',padding = 'same'))
model15.add(Conv2D(64, (3, 3), activation='relu',kernel_initializer='he_normal',padding = 'same'))
model15.add(MaxPooling2D((2, 2)))
model15.add(Flatten())
model15.add(Dense(128, activation='relu'))
model15.add(Dropout(rate = 0.6))
model15.add(Dense(10, activation='softmax'))
model15.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 [7]:
```
epochs = 10
model15.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
model15.fit(X_train,y_train,epochs=epochs,batch_size = 32)
```

```
Epoch 1/10
50000/50000 [==============================] - 463s 9ms/step - loss: 1.6815 - acc: 0.3767
Epoch 2/10
50000/50000 [==============================] - 463s 9ms/step - loss: 1.1605 - acc: 0.5887
Epoch 3/10
50000/50000 [==============================] - 463s 9ms/step - loss: 0.9534 - acc: 0.6736
Epoch 4/10
50000/50000 [==============================] - 456s 9ms/step - loss: 0.8250 - acc: 0.7199
Epoch 5/10
50000/50000 [==============================] - 457s 9ms/step - loss: 0.7360 - acc: 0.7537
Epoch 6/10
50000/50000 [==============================] - 472s 9ms/step - loss: 0.6739 - acc: 0.7756
Epoch 7/10
50000/50000 [==============================] - 449s 9ms/step - loss: 0.6259 - acc: 0.7907
Epoch 8/10
50000/50000 [==============================] - 378s 8ms/step - loss: 0.5763 - acc: 0.8080
Epoch 9/10
50000/50000 [==============================] - 362s 7ms/step - loss: 0.5425 - acc: 0.8201
Epoch 10/10
50000/50000 [==============================] - 376s 8ms/step - loss: 0.5054 - acc: 0.8319
```

Out[7]:
```
<keras.callbacks.History at 0x2513a0cab38>
```

In [8]:
```
test_loss,test_acc = model15.evaluate(X_test,y_test)
test_acc
```

```
10000/10000 [==============================] - 38s 4ms/step
```

Out[8]:
```
0.7611
```

Model 16

optimizer - adam with 0.7 dropout rate

In [10]:
```
model16 = Sequential()
model16.add(Conv2D(64, (3, 3), activation='relu',kernel_initializer='he_normal',padding = 'same', input_shape=(32, 32, 3)))
model16.add(Conv2D(64, (3, 3), activation='relu',kernel_initializer='he_normal',padding = 'same'))
model16.add(MaxPooling2D((2, 2)))
model16.add(Conv2D(64, (3, 3), activation='relu',kernel_initializer='he_normal',padding = 'same'))
model16.add(Conv2D(64, (3, 3), activation='relu',kernel_initializer='he_normal',padding = 'same'))
model16.add(MaxPooling2D((2, 2)))
model16.add(Conv2D(64, (3, 3), activation='relu',kernel_initializer='he_normal',padding = 'same'))
model16.add(Conv2D(64, (3, 3), activation='relu',kernel_initializer='he_normal',padding = 'same'))
model16.add(MaxPooling2D((2, 2)))
model16.add(Flatten())
model16.add(Dense(128, activation='relu'))
model16.add(Dropout(rate = 0.7))
model16.add(Dense(10, activation='softmax'))
model16.summary()
```

Layer (type)	Output Shape	Param #
conv2d_22 (Conv2D)	(None, 32, 32, 64)	1792
conv2d_23 (Conv2D)	(None, 32, 32, 64)	36928
max_pooling2d_16 (MaxPooling	(None, 16, 16, 64)	0
conv2d_24 (Conv2D)	(None, 16, 16, 64)	36928
conv2d_25 (Conv2D)	(None, 16, 16, 64)	36928
max_pooling2d_17 (MaxPooling	(None, 8, 8, 64)	0
conv2d_26 (Conv2D)	(None, 8, 8, 64)	36928
max_pooling2d_18 (MaxPooling	(None, 4, 4, 64)	0
conv2d_27 (Conv2D)	(None, 4, 4, 64)	36928
max_pooling2d_19 (MaxPooling	(None, 2, 2, 64)	0
conv2d_28 (Conv2D)	(None, 2, 2, 64)	36928
max_pooling2d_20 (MaxPooling	(None, 1, 1, 64)	0
flatten_4 (Flatten)	(None, 64)	0
dense_7 (Dense)	(None, 128)	8320
dropout_4 (Dropout)	(None, 128)	0
dense_8 (Dense)	(None, 10)	1290

```
Total params: 232,970
Trainable params: 232,970
Non-trainable params: 0
_________________________________________________________________
```

In [11]:
```
epochs = 10
model16.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
model16.fit(X_train,y_train,epochs=epochs,batch_size = 32)
```

```
Epoch 1/10
50000/50000 [==============================] - 347s 7ms/step - loss: 1.7212 - acc: 0.3608
Epoch 2/10
50000/50000 [==============================] - 391s 8ms/step - loss: 1.2334 - acc: 0.5646
Epoch 3/10
50000/50000 [==============================] - 370s 7ms/step - loss: 1.0350 - acc: 0.6443
Epoch 4/10
50000/50000 [==============================] - 394s 8ms/step - loss: 0.9073 - acc: 0.6932
Epoch 5/10
50000/50000 [==============================] - 351s 7ms/step - loss: 0.8090 - acc: 0.7313
Epoch 6/10
50000/50000 [==============================] - 352s 7ms/step - loss: 0.7399 - acc: 0.7558
Epoch 7/10
50000/50000 [==============================] - 377s 8ms/step - loss: 0.6818 - acc: 0.7762
Epoch 8/10
50000/50000 [==============================] - 396s 8ms/step - loss: 0.6335 - acc: 0.7906
Epoch 9/10
50000/50000 [==============================] - 349s 7ms/step - loss: 0.5927 - acc: 0.8048
Epoch 10/10
50000/50000 [==============================] - 404s 8ms/step - loss: 0.5594 - acc: 0.8154
```

Out[11]:
```
<keras.callbacks.History at 0x2517189fe48>
```

In [12]:
```
test_loss,test_acc = model16.evaluate(X_test,y_test)
test_acc
```

```
10000/10000 [==============================] - 31s 3ms/step
```

Out[12]:
```
0.7503
```

Observation

The optimizer adam and sgd almost performing the same and the overfitting is controlled by the selection of dropout rate.