Model performance on Dropout

- Dropout is a technique where randomly selected neurons are ignored during training. They are "dropped-out" randomly. This means that their contribution to the activation of downstream neurons is temporally removed on the forward pass and any weight updates are not applied to the neuron on the backward pass.

Dropout Rate

The default interpretation of the dropout hyperparameter is the probability of training a given node in a layer, where 1.0 means no dropout, and 0.0 means no outputs from the layer.

A good value for dropout in a hidden layer is between 0.5 and 0.8. Input layers use a larger dropout rate, such as of 0.8.

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, BatchNormalisation
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)
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 11

Dropout rate - 0.5

In [3]:
```
model11 = Sequential()
model11.add(Conv2D(64, (3, 3), activation='relu',kernel_initializer='he_normal',kernel_regularizer=v
egularizers.l2(0.001),padding = 'same', input_shape=(32, 32, 3)))
model11.add(Conv2D(64, (3, 3), activation='relu',kernel_initializer='he_normal',kernel_regularizer=v
egularizers.l2(0.001),padding = 'same'))
model11.add(MaxPooling2D((2, 2)))
model11.add(Conv2D(64, (3, 3), activation='relu',kernel_initializer='he_normal',kernel_regularizer=v
egularizers.l2(0.001),padding = 'same'))
model11.add(Conv2D(64, (3, 3), activation='relu',kernel_initializer='he_normal',kernel_regularizer=v
egularizers.l2(0.001),padding = 'same'))
model11.add(MaxPooling2D((2, 2)))
model11.add(Conv2D(64, (3, 3), activation='relu',kernel_initializer='he_normal',kernel_regularizer=v
egularizers.l2(0.001),padding = 'same'))
model11.add(MaxPooling2D((2, 2)))
model11.add(Flatten())
model11.add(Dense(128, activation='relu'))
model11.add(Dropout(rate = 0.5))
model11.add(Dense(10, activation='softmax'))
model11.summary()
```

Layer (type)	Output Shape	Param #
conv2d_6 (Conv2D)	(None, 32, 32, 64)	1792
conv2d_7 (Conv2D)	(None, 32, 32, 64)	36928
max_pooling2d_4 (MaxPooling2	(None, 16, 16, 64)	0
conv2d_8 (Conv2D)	(None, 16, 16, 64)	36928
conv2d_9 (Conv2D)	(None, 16, 16, 64)	36928
max_pooling2d_5 (MaxPooling2	(None, 8, 8, 64)	0
conv2d_10 (Conv2D)	(None, 8, 8, 64)	36928
max_pooling2d_6 (MaxPooling2	(None, 4, 4, 64)	0
flatten_2 (Flatten)	(None, 1024)	0
dense_3 (Dense)	(None, 128)	131200
dropout_2 (Dropout)	(None, 128)	0
dense_4 (Dense)	(None, 10)	1290

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

In [4]:
```
epochs = 10
sgd = SGD(lr=1e-2, momentum=0.9, decay=1e-2/epochs)
model11.compile(optimizer=sgd, loss='categorical_crossentropy', metrics=['accuracy'])
model11.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 [==============================] - 462s 9ms/step - loss: 2.3279 - acc: 0.3312
Epoch 2/10
50000/50000 [==============================] - 454s 9ms/step - loss: 1.8172 - acc: 0.4952
Epoch 3/10
50000/50000 [==============================] - 457s 9ms/step - loss: 1.5936 - acc: 0.5647
Epoch 4/10
50000/50000 [==============================] - 459s 9ms/step - loss: 1.4546 - acc: 0.6104
Epoch 5/10
50000/50000 [==============================] - 453s 9ms/step - loss: 1.3538 - acc: 0.6418
Epoch 6/10
50000/50000 [==============================] - 445s 9ms/step - loss: 1.2824 - acc: 0.6627
Epoch 7/10
50000/50000 [==============================] - 435s 9ms/step - loss: 1.2316 - acc: 0.6786
Epoch 8/10
50000/50000 [==============================] - 427s 9ms/step - loss: 1.1815 - acc: 0.6928
Epoch 9/10
50000/50000 [==============================] - 443s 9ms/step - loss: 1.1435 - acc: 0.7023
Epoch 10/10
50000/50000 [==============================] - 463s 9ms/step - loss: 1.1068 - acc: 0.7148
```

Out[4]:
```
<keras.callbacks.History at 0x26997724e48>
```

In [6]:
```
test_loss,test_acc = model11.evaluate(X_test,y_test)
test_acc
```

```
10000/10000 [==============================] - 37s 4ms/step
```

Out[6]:
```
0.7262
```

Observation

In the model 11 dropout reduce the overfitting

model 12

Add one more layer of conv2d and maxpooling and remove kernel_regularizer

In [7]:
```
model12 = Sequential()
model12.add(Conv2D(64, (3, 3), activation='relu',kernel_initializer='he_normal',padding = 'same', in
put_shape=(32, 32, 3)))
model12.add(Conv2D(64, (3, 3), activation='relu',kernel_initializer='he_normal',padding = 'same'))
model12.add(MaxPooling2D((2, 2)))
model12.add(Conv2D(64, (3, 3), activation='relu',kernel_initializer='he_normal',padding = 'same'))
model12.add(Conv2D(64, (3, 3), activation='relu',kernel_initializer='he_normal',padding = 'same'))
model12.add(MaxPooling2D((2, 2)))
model12.add(Conv2D(64, (3, 3), activation='relu',kernel_initializer='he_normal',padding = 'same'))
model12.add(MaxPooling2D((2, 2)))
model12.add(Conv2D(64, (3, 3), activation='relu',kernel_initializer='he_normal',padding = 'same'))
model12.add(MaxPooling2D((2, 2)))
model12.add(Flatten())
model12.add(Dense(128, activation='relu'))
model12.add(Dropout(rate = 0.5))
model12.add(Dense(10, activation='softmax'))
model12.summary()
```

Layer (type)	Output Shape	Param #
conv2d_11 (Conv2D)	(None, 32, 32, 64)	1792
conv2d_12 (Conv2D)	(None, 32, 32, 64)	36928
max_pooling2d_7 (MaxPooling2	(None, 16, 16, 64)	0
conv2d_13 (Conv2D)	(None, 16, 16, 64)	36928
conv2d_14 (Conv2D)	(None, 16, 16, 64)	36928
max_pooling2d_8 (MaxPooling2	(None, 8, 8, 64)	0
conv2d_15 (Conv2D)	(None, 8, 8, 64)	36928
max_pooling2d_9 (MaxPooling2	(None, 4, 4, 64)	0
conv2d_16 (Conv2D)	(None, 4, 4, 64)	36928
max_pooling2d_10 (MaxPooling	(None, 2, 2, 64)	0
flatten_3 (Flatten)	(None, 256)	0
dense_5 (Dense)	(None, 128)	32896
dropout_3 (Dropout)	(None, 128)	0
dense_6 (Dense)	(None, 10)	1290

```
Total params: 220,618
Trainable params: 220,618
Non-trainable params: 0
```

In [8]:
```
epochs = 10
sgd = SGD(lr=1e-2, momentum=0.9, decay=1e-2/epochs)
model12.compile(optimizer=sgd, loss='categorical_crossentropy', metrics=['accuracy'])
model12.fit(X_train,y_train,epochs=epochs,batch_size = 32)
```

```
Epoch 1/10
50000/50000 [==============================] - 435s 9ms/step - loss: 1.7725 - acc: 0.3398
Epoch 2/10
50000/50000 [==============================] - 431s 9ms/step - loss: 1.3314 - acc: 0.5141
Epoch 3/10
50000/50000 [==============================] - 430s 9ms/step - loss: 1.1408 - acc: 0.5899
Epoch 4/10
50000/50000 [==============================] - 428s 9ms/step - loss: 1.0232 - acc: 0.6321
Epoch 5/10
50000/50000 [==============================] - 429s 9ms/step - loss: 0.9456 - acc: 0.6619
Epoch 6/10
50000/50000 [==============================] - 431s 9ms/step - loss: 0.8827 - acc: 0.6877
Epoch 7/10
50000/50000 [==============================] - 419s 8ms/step - loss: 0.8348 - acc: 0.7034
Epoch 8/10
50000/50000 [==============================] - 361s 7ms/step - loss: 0.7932 - acc: 0.7178
Epoch 9/10
50000/50000 [==============================] - 391s 8ms/step - loss: 0.7533 - acc: 0.7351
Epoch 10/10
50000/50000 [==============================] - 376s 8ms/step - loss: 0.7236 - acc: 0.7455
```

Out[8]:
```
<keras.callbacks.History at 0x2699772a748>
```

In [9]:
```
test_loss,test_acc = model12.evaluate(X_test,y_test)
test_acc
```

```
10000/10000 [==============================] - 30s 3ms/step
```

Out[9]:
```
0.7259
```

Observation

The addition of one hidden layer increased the accuracy by 3 % in same 10 epochs

Model 13

Dropout rate 0.6

In [11]:
```
model13 = Sequential()
model13.add(Conv2D(64, (3, 3), activation='relu',kernel_initializer='he_normal',padding = 'same', in
put_shape=(32, 32, 3)))
model13.add(Conv2D(64, (3, 3), activation='relu',kernel_initializer='he_normal',padding = 'same'))
model13.add(MaxPooling2D((2, 2)))
model13.add(Conv2D(64, (3, 3), activation='relu',kernel_initializer='he_normal',padding = 'same'))
model13.add(Conv2D(64, (3, 3), activation='relu',kernel_initializer='he_normal',padding = 'same'))
model13.add(MaxPooling2D((2, 2)))
model13.add(Conv2D(64, (3, 3), activation='relu',kernel_initializer='he_normal',padding = 'same'))
model13.add(MaxPooling2D((2, 2)))
model13.add(Conv2D(64, (3, 3), activation='relu',kernel_initializer='he_normal',padding = 'same'))
model13.add(MaxPooling2D((2, 2)))
model13.add(Flatten())
model13.add(Dense(128, activation='relu'))
model13.add(Dropout(rate = 0.6))
model13.add(Dense(10, activation='softmax'))
model13.summary()
```

Layer (type)	Output Shape	Param #
conv2d_24 (Conv2D)	(None, 32, 32, 64)	1792
conv2d_25 (Conv2D)	(None, 32, 32, 64)	36928
max_pooling2d_15 (MaxPooling	(None, 16, 16, 64)	0
conv2d_26 (Conv2D)	(None, 16, 16, 64)	36928
conv2d_27 (Conv2D)	(None, 16, 16, 64)	36928
max_pooling2d_16 (MaxPooling	(None, 8, 8, 64)	0
conv2d_28 (Conv2D)	(None, 8, 8, 64)	36928
max_pooling2d_17 (MaxPooling	(None, 4, 4, 64)	0
conv2d_29 (Conv2D)	(None, 4, 4, 64)	36928
max_pooling2d_18 (MaxPooling	(None, 2, 2, 64)	0
conv2d_30 (Conv2D)	(None, 2, 2, 64)	36928
max_pooling2d_19 (MaxPooling	(None, 1, 1, 64)	0
flatten_5 (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 [12]:
```
epochs = 10
sgd = SGD(lr=1e-2, momentum=0.9, decay=1e-2/epochs)
model13.compile(optimizer=sgd, loss='categorical_crossentropy', metrics=['accuracy'])
model13.fit(X_train,y_train,epochs=epochs,batch_size = 32)
```

```
Epoch 1/10
50000/50000 [==============================] - 370s 7ms/step - loss: 1.8009 - acc: 0.3238
Epoch 2/10
50000/50000 [==============================] - 393s 8ms/step - loss: 1.3372 - acc: 0.5133
Epoch 3/10
50000/50000 [==============================] - 347s 7ms/step - loss: 1.1372 - acc: 0.5952
Epoch 4/10
50000/50000 [==============================] - 354s 7ms/step - loss: 1.0139 - acc: 0.6429
Epoch 5/10
50000/50000 [==============================] - 346s 7ms/step - loss: 0.9252 - acc: 0.6763
Epoch 6/10
50000/50000 [==============================] - 349s 7ms/step - loss: 0.8498 - acc: 0.7057
Epoch 7/10
50000/50000 [==============================] - 354s 7ms/step - loss: 0.7909 - acc: 0.7255
Epoch 8/10
50000/50000 [==============================] - 349s 7ms/step - loss: 0.7416 - acc: 0.7459
Epoch 9/10
50000/50000 [==============================] - 349s 7ms/step - loss: 0.7006 - acc: 0.7618
Epoch 10/10
50000/50000 [==============================] - 348s 7ms/step - loss: 0.6598 - acc: 0.7740
```

Out[12]:
```
<keras.callbacks.History at 0x249d9c6d5f8>
```

In [13]:
```
test_loss,test_acc = model13.evaluate(X_test,y_test)
test_acc
```

```
10000/10000 [==============================] - 29s 3ms/step
```

Out[13]:
```
0.7357
```

Observation

The model 13 gives more accuracy then other models. Next will tune hyperparmeters like batch size, optimizer, loss.