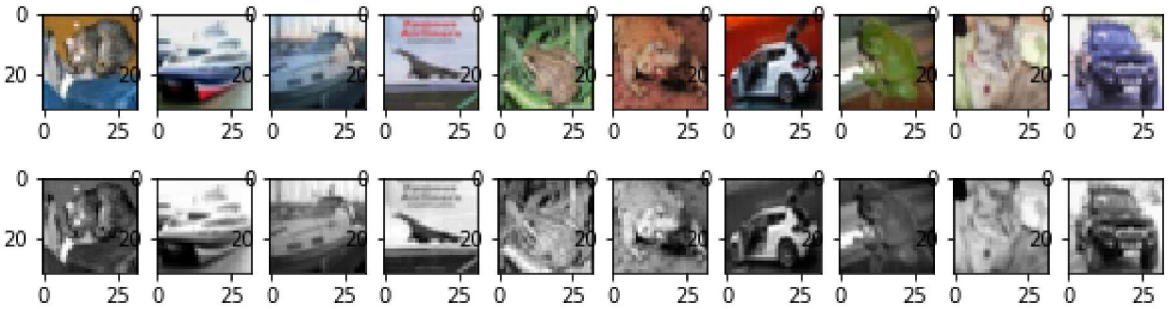
Howework 2: Classification of Cifar-10 images using Keras

Name: Akwasi Darkwah Akwaboah

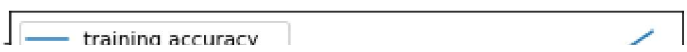
Large Network (8 layers) 512 nodes each

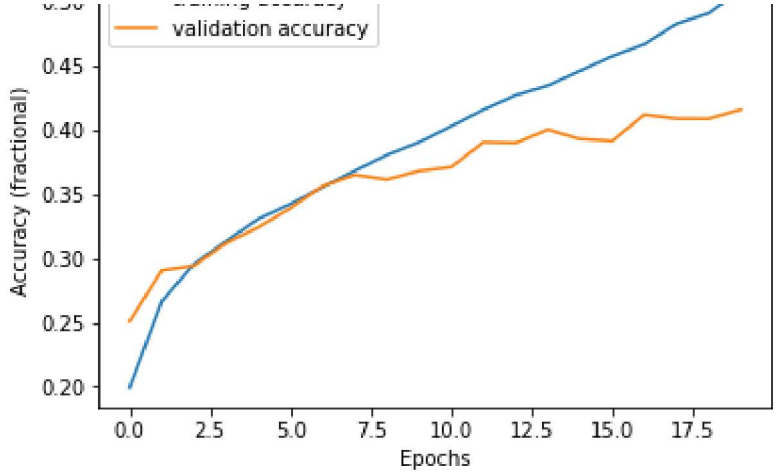
```
#implementing in Keras
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
from tensorflow import keras

#(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()

#loading dataset
from keras.datasets import cifar10
(x_train, y_train), (x_test, y_test) = cifar10.load_data()

#1. Plotting the first 10 cifar10 images
# fig, axes = plt.subplots(1, 10, figsize=(10,5))
# for img, ax in zip(x_train[:10], axes):
#   ax.imshow(img)
# plt.show()

#2. Convert xtest and xtrain to float32 and normalize data to range 0.0 - 1.0
x_train_norm = np.float32(x_train/255.0)
x_test_norm = np.float32(x_test/255.0)
#print(x_train_norm)
#print(x_test_norm)

#3. function to convert to grayscale
def grayscale(data, dtype='float32'):
  # luma coding weighted average in video systems
  r, g, b = np.asarray(.3, dtype=dtype), np.asarray(.59, dtype=dtype), np.asarray(.11, dtype=dtype)
  rst = r * data[:, :, :, 0] + g * data[:, :, :, 1] + b * data[:, :, :, 2]
  # add channel dimension
  rst = np.expand_dims(rst, axis=3)
  return rst

#function Call
X_train_gray = grayscale(x_train_norm)
X_test_gray = grayscale(x_test_norm)


#plot a few grayscale vrs rgb
fig, axes = plt.subplots(1, 10, figsize=(10,5))
for img, ax in zip(x_test[:10], axes):
  ax.imshow(img)
plt.show()

fig, axes = plt.subplots(1, 10, figsize=(10,5))
for img, ax in zip(X_test_gray[:10], axes):
  ax.imshow(img[:,:,0], cmap=plt.get_cmap('gray'), interpolation='none')
plt.show()

#create ANN model
#model constants

#define network
model = keras.Sequential([
    keras.layers.Flatten(input_shape = (32, 32, 1)),
    keras.layers.Dense(512, activation = 'relu'),
    keras.layers.Dense(512, activation = 'relu'),
    keras.layers.Dense(512, activation = 'relu'),
    keras.layers.Dense(512, activation = 'relu'),
```

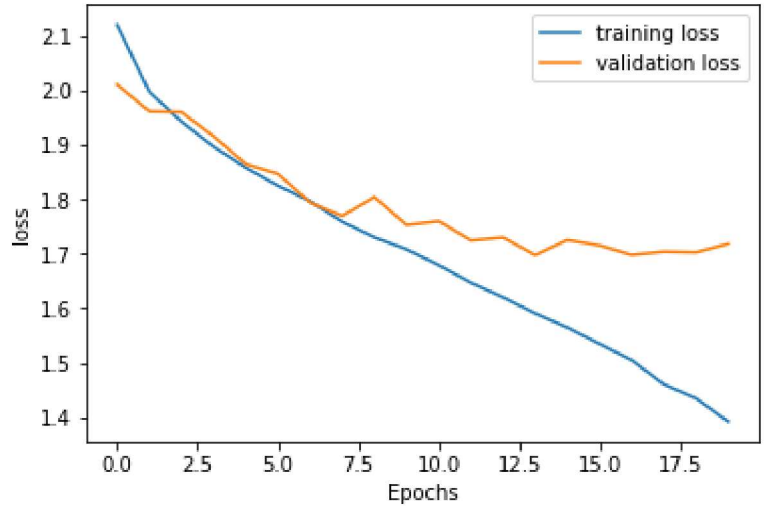
```
        keras.layers.Dense(512, activation = 'relu'),
        keras.layers.Dense(512, activation = 'relu'),
        keras.layers.Dense(512, activation = 'relu'),
        keras.layers.Dense(512, activation = 'relu'),
        keras.layers.Dense(10, activation = 'softmax')
    ])

    #Compile Network
    model.compile(optimizer='adam', loss ='sparse_categorical_crossentropy',
                  metrics = ['accuracy'])

    #train the network with 20% validation
    history = model.fit(X_train_gray, y_train, batch_size = 128, epochs=20, validation_split=0.2) #spits

    #test accuracy and loss
    test_loss, test_acc = model.evaluate(X_test_gray, y_test, verbose=2)
    print('\nTest accuracy:', test_acc)

    predictions = model.predict(X_test_gray)

    predictions[0]

    np.argmax(predictions[0])

    print(history.history.keys())
    plt.plot(history.history['acc'])
    plt.plot(history.history['val_acc'])
    plt.xlabel('Epochs')
    plt.ylabel('Accuracy (fractional)')
    plt.legend(['training accuracy', 'validation accuracy'], loc='best')

    plt.figure()
    plt.plot(history.history['loss'])
    plt.plot(history.history['val_loss'])
    plt.xlabel('Epochs')
    plt.ylabel('loss')
    plt.legend(['training loss', 'validation loss'], loc='best')
```



```
Train on 40000 samples, validate on 10000 samples
Epoch 1/20
40000/40000 [==============================] - 19s 482us/sample - loss: 2.1207 - acc: 0.
Epoch 2/20
40000/40000 [==============================] - 19s 472us/sample - loss: 1.9978 - acc: 0.
Epoch 3/20
40000/40000 [==============================] - 19s 474us/sample - loss: 1.9430 - acc: 0.
Epoch 4/20
40000/40000 [==============================] - 19s 469us/sample - loss: 1.8967 - acc: 0.
Epoch 5/20
40000/40000 [==============================] - 19s 472us/sample - loss: 1.8584 - acc: 0.
Epoch 6/20
40000/40000 [==============================] - 19s 473us/sample - loss: 1.8252 - acc: 0.
Epoch 7/20
40000/40000 [==============================] - 19s 468us/sample - loss: 1.7967 - acc: 0.
Epoch 8/20
40000/40000 [==============================] - 19s 470us/sample - loss: 1.7590 - acc: 0.
Epoch 9/20
40000/40000 [==============================] - 19s 472us/sample - loss: 1.7303 - acc: 0.
Epoch 10/20
40000/40000 [==============================] - 19s 468us/sample - loss: 1.7082 - acc: 0.
Epoch 11/20
40000/40000 [==============================] - 19s 470us/sample - loss: 1.6792 - acc: 0.
Epoch 12/20
40000/40000 [==============================] - 19s 480us/sample - loss: 1.6469 - acc: 0.
Epoch 13/20
40000/40000 [==============================] - 19s 469us/sample - loss: 1.6201 - acc: 0.
Epoch 14/20
40000/40000 [==============================] - 19s 468us/sample - loss: 1.5905 - acc: 0.
Epoch 15/20
40000/40000 [==============================] - 19s 469us/sample - loss: 1.5645 - acc: 0.
Epoch 16/20
40000/40000 [==============================] - 19s 469us/sample - loss: 1.5344 - acc: 0.
Epoch 17/20
40000/40000 [==============================] - 19s 477us/sample - loss: 1.5048 - acc: 0.
Epoch 18/20
40000/40000 [==============================] - 19s 470us/sample - loss: 1.4601 - acc: 0.
Epoch 19/20
40000/40000 [==============================] - 19s 474us/sample - loss: 1.4345 - acc: 0.
Epoch 20/20
40000/40000 [==============================] - 19s 472us/sample - loss: 1.3916 - acc: 0.
10000/10000 - 2s - loss: 1.7181 - acc: 0.4150

Test accuracy: 0.415
dict_keys(['loss', 'acc', 'val_loss', 'val_acc'])
<matplotlib.legend.Legend at 0x7f781d9aaba8>
```



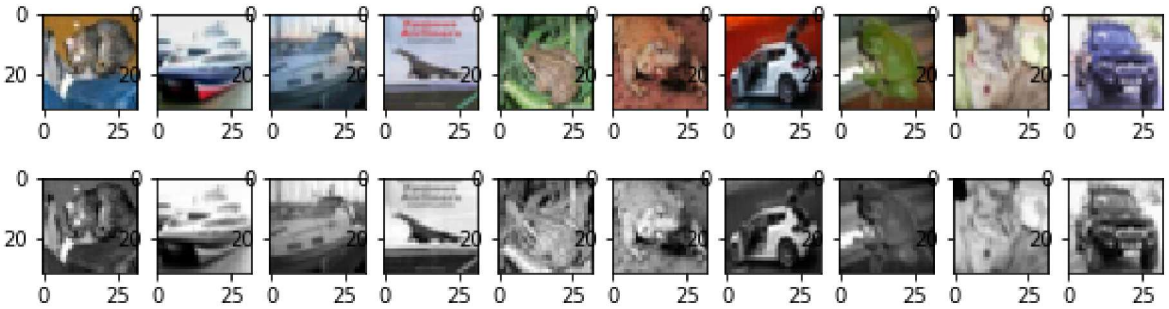




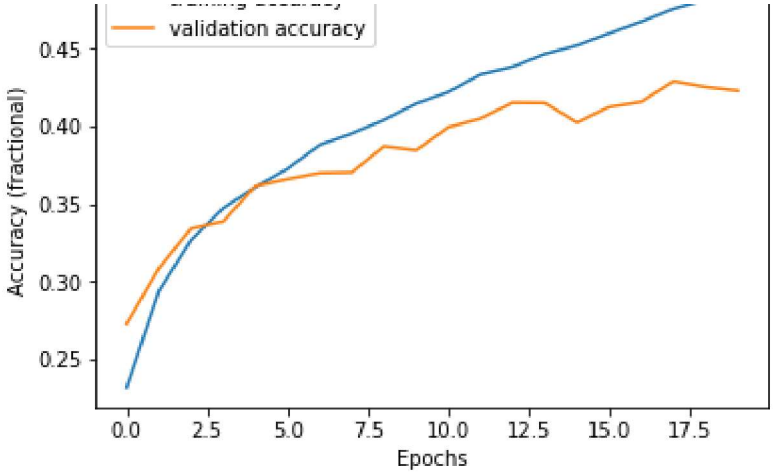
Smaller network: 4-layers (tapering 512-256-64-32)

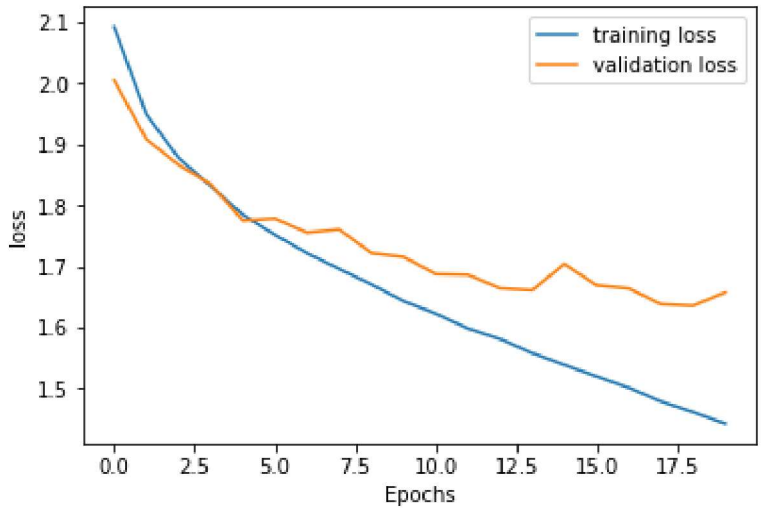
```
#implementing in Keras
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
from tensorflow import keras

#(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()

#loading dataset
from keras.datasets import cifar10
(x_train, y_train), (x_test, y_test) = cifar10.load_data()

#1. Plotting the first 10 cifar10 images
# fig, axes = plt.subplots(1, 10, figsize=(10,5))
# for img, ax in zip(x_train[:10], axes):
#   ax.imshow(img)
# plt.show()

#2. Convert xtest and xtrain to float32 and normalize data to range 0.0 - 1.0
x_train_norm = np.float32(x_train/255.0)
x_test_norm = np.float32(x_test/255.0)
#print(x_train_norm)
#print(x_test_norm)

#3. function to convert to grayscale
def grayscale(data, dtype='float32'):
  # luma coding weighted average in video systems
```

```
    r, g, b = np.asarray(.3, dtype=dtype), np.asarray(.59, dtype=dtype), np.asarray(.11, dtype=dtype)
    rst = r * data[:, :, :, 0] + g * data[:, :, :, 1] + b * data[:, :, :, 2]
    # add channel dimension
    rst = np.expand_dims(rst, axis=3)
    return rst

#function Call
X_train_gray = grayscale(x_train_norm)
X_test_gray = grayscale(x_test_norm)


#plot a few grayscale vrs rgb
fig, axes = plt.subplots(1, 10, figsize=(10,5))
for img, ax in zip(x_test[:10], axes):
  ax.imshow(img)
plt.show()

fig, axes = plt.subplots(1, 10, figsize=(10,5))
for img, ax in zip(X_test_gray[:10], axes):
  ax.imshow(img[:,:,0], cmap=plt.get_cmap('gray'), interpolation='none')
plt.show()

#create ANN model
#model constants

#define network
model = keras.Sequential([
    keras.layers.Flatten(input_shape = (32, 32, 1)),
    keras.layers.Dense(512, activation = 'relu'),
    keras.layers.Dense(256, activation = 'relu'),
    keras.layers.Dense(64, activation = 'relu'),
    keras.layers.Dense(32, activation = 'relu'),
    keras.layers.Dense(10, activation = 'softmax')
])

#Compile Network
model.compile(optimizer='adam', loss ='sparse_categorical_crossentropy',
              metrics = ['accuracy'])

#train the network with 20% validation
history = model.fit(X_train_gray, y_train, batch_size = 128, epochs=20, validation_split=0.2) #spits

#test accuracy and loss
test_loss, test_acc = model.evaluate(X_test_gray, y_test, verbose=2)
print('\nTest accuracy:', test_acc)

predictions = model.predict(X_test_gray)

predictions[0]

np.argmax(predictions[0])

print(history.history.keys())
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.xlabel('Epochs')
plt.ylabel('Accuracy (fractional)')
plt.legend(['training accuracy', 'validation accuracy'], loc='best')

plt.figure()
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.xlabel('Epochs')
plt.ylabel('loss')
plt.legend(['training loss', 'validation loss'], loc='best')
```



```
Train on 40000 samples, validate on 10000 samples
Epoch 1/20
40000/40000 [==============================] - 6s 138us/sample - loss: 2.0931 - acc: 0.2
Epoch 2/20
40000/40000 [==============================] - 5s 127us/sample - loss: 1.9495 - acc: 0.2
Epoch 3/20
40000/40000 [==============================] - 5s 126us/sample - loss: 1.8782 - acc: 0.3
Epoch 4/20
40000/40000 [==============================] - 5s 127us/sample - loss: 1.8320 - acc: 0.3
Epoch 5/20
40000/40000 [==============================] - 5s 124us/sample - loss: 1.7848 - acc: 0.3
Epoch 6/20
40000/40000 [==============================] - 5s 123us/sample - loss: 1.7516 - acc: 0.3
Epoch 7/20
40000/40000 [==============================] - 5s 122us/sample - loss: 1.7216 - acc: 0.3
Epoch 8/20
40000/40000 [==============================] - 5s 123us/sample - loss: 1.6957 - acc: 0.3
Epoch 9/20
40000/40000 [==============================] - 5s 122us/sample - loss: 1.6703 - acc: 0.4
Epoch 10/20
40000/40000 [==============================] - 5s 123us/sample - loss: 1.6429 - acc: 0.4
Epoch 11/20
40000/40000 [==============================] - 5s 122us/sample - loss: 1.6223 - acc: 0.4
Epoch 12/20
40000/40000 [==============================] - 5s 122us/sample - loss: 1.5978 - acc: 0.4
Epoch 13/20
40000/40000 [==============================] - 5s 122us/sample - loss: 1.5805 - acc: 0.4
Epoch 14/20
40000/40000 [==============================] - 5s 123us/sample - loss: 1.5576 - acc: 0.4
Epoch 15/20
40000/40000 [==============================] - 5s 121us/sample - loss: 1.5381 - acc: 0.4
Epoch 16/20
40000/40000 [==============================] - 5s 123us/sample - loss: 1.5192 - acc: 0.4
Epoch 17/20
40000/40000 [==============================] - 5s 126us/sample - loss: 1.5006 - acc: 0.4
Epoch 18/20
40000/40000 [==============================] - 5s 126us/sample - loss: 1.4781 - acc: 0.4
Epoch 19/20
40000/40000 [==============================] - 5s 124us/sample - loss: 1.4606 - acc: 0.4
Epoch 20/20
40000/40000 [==============================] - 5s 123us/sample - loss: 1.4416 - acc: 0.4
10000/10000 - 1s - loss: 1.6674 - acc: 0.4151

Test accuracy: 0.4151
dict_keys(['loss', 'acc', 'val_loss', 'val_acc'])
<matplotlib.legend.Legend at 0x7f78170acef0>
```

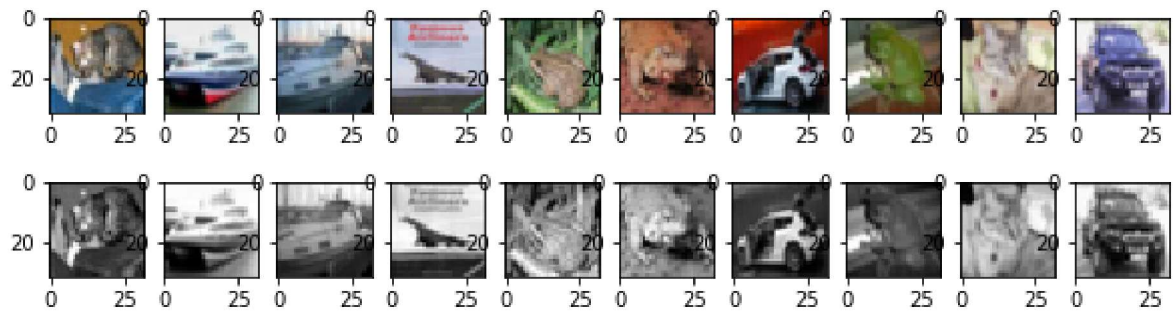




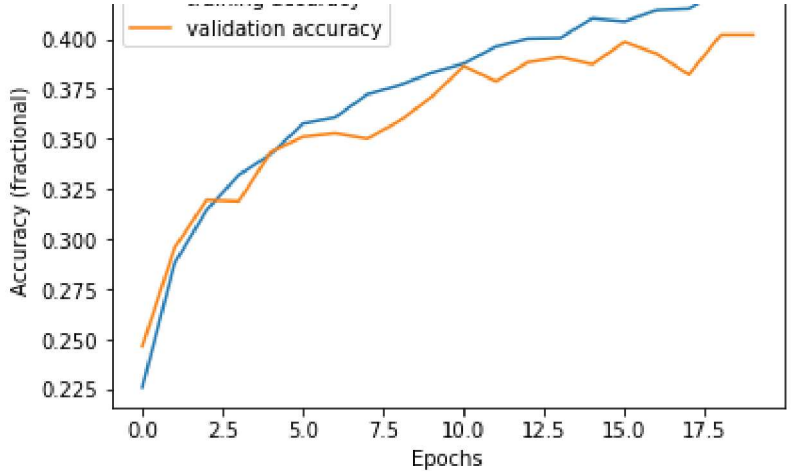
L2-norm regularization

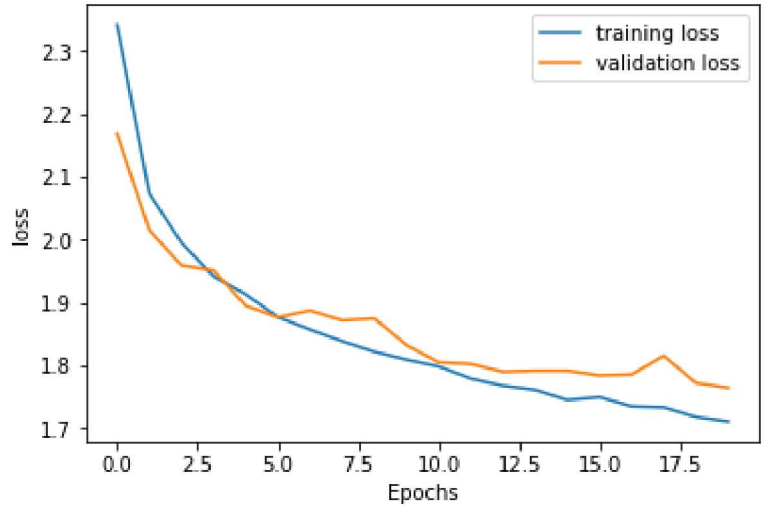
```
#implementing in Keras
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
from tensorflow import keras
from keras import regularizers

#(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()

#loading dataset
from keras.datasets import cifar10
(x_train, y_train), (x_test, y_test) = cifar10.load_data()

#1. Plotting the first 10 cifar10 images
# fig, axes = plt.subplots(1, 10, figsize=(10,5))
# for img, ax in zip(x_train[:10], axes):
#   ax.imshow(img)
# plt.show()

#2. Convert xtest and xtrain to float32 and normalize data to range 0.0 - 1.0
x_train_norm = np.float32(x_train/255.0)
x_test_norm = np.float32(x_test/255.0)
#print(x_train_norm)
#print(x_test_norm)

#3. function to convert to grayscale
def grayscale(data, dtype='float32'):
```

```
    # luma coding weighted average in video systems
    r, g, b = np.asarray(.3, dtype=dtype), np.asarray(.59, dtype=dtype), np.asarray(.11, dtype=dtype)
    rst = r * data[:, :, :, 0] + g * data[:, :, :, 1] + b * data[:, :, :, 2]
    # add channel dimension
    rst = np.expand_dims(rst, axis=3)
    return rst

#function Call
X_train_gray = grayscale(x_train_norm)
X_test_gray = grayscale(x_test_norm)


#plot a few grayscale vrs rgb
fig, axes = plt.subplots(1, 10, figsize=(10,5))
for img, ax in zip(x_test[:10], axes):
  ax.imshow(img)
plt.show()

fig, axes = plt.subplots(1, 10, figsize=(10,5))
for img, ax in zip(X_test_gray[:10], axes):
  ax.imshow(img[:,:,0], cmap=plt.get_cmap('gray'), interpolation='none')
plt.show()

#create ANN model
#model constants

#define network
l2_alpha = 0.0005

model = keras.Sequential([
    keras.layers.Flatten(input_shape = (32, 32, 1)),
    keras.layers.Dense(512, activation = 'relu', kernel_regularizer=regularizers.l2(l2_alpha)),
    keras.layers.Dense(256, activation = 'relu', kernel_regularizer=regularizers.l2(l2_alpha)),
    keras.layers.Dense(64, activation = 'relu', kernel_regularizer=regularizers.l2(l2_alpha)),
    keras.layers.Dense(32, activation = 'relu', kernel_regularizer=regularizers.l2(l2_alpha)),
    keras.layers.Dense(10, activation = 'softmax')
])

#Compile Network
model.compile(optimizer='adam', loss ='sparse_categorical_crossentropy',
              metrics = ['accuracy'])

#train the network with 20% validation
history = model.fit(X_train_gray, y_train, batch_size = 128, epochs=20, validation_split=0.2) #spits

#test accuracy and loss
test_loss, test_acc = model.evaluate(X_test_gray, y_test, verbose=2)
print('\nTest accuracy:', test_acc)

predictions = model.predict(X_test_gray)

predictions[0]

np.argmax(predictions[0])

print(history.history.keys())
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.xlabel('Epochs')
plt.ylabel('Accuracy (fractional)')
plt.legend(['training accuracy', 'validation accuracy'], loc='best')

plt.figure()
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.xlabel('Epochs')
plt.ylabel('loss')
plt.legend(['training loss', 'validation loss'], loc='best')
```



```
Train on 40000 samples, validate on 10000 samples
Epoch 1/20
40000/40000 [==============================] - 6s 153us/sample - loss: 2.3419 - acc: 0.2
Epoch 2/20
40000/40000 [==============================] - 5s 137us/sample - loss: 2.0735 - acc: 0.2
Epoch 3/20
40000/40000 [==============================] - 5s 137us/sample - loss: 1.9953 - acc: 0.3
Epoch 4/20
40000/40000 [==============================] - 6s 139us/sample - loss: 1.9417 - acc: 0.3
Epoch 5/20
40000/40000 [==============================] - 6s 140us/sample - loss: 1.9130 - acc: 0.3
Epoch 6/20
40000/40000 [==============================] - 6s 138us/sample - loss: 1.8769 - acc: 0.3
Epoch 7/20
40000/40000 [==============================] - 6s 141us/sample - loss: 1.8569 - acc: 0.3
Epoch 8/20
40000/40000 [==============================] - 6s 138us/sample - loss: 1.8384 - acc: 0.3
Epoch 9/20
40000/40000 [==============================] - 6s 138us/sample - loss: 1.8217 - acc: 0.3
Epoch 10/20
40000/40000 [==============================] - 5s 137us/sample - loss: 1.8090 - acc: 0.3
Epoch 11/20
40000/40000 [==============================] - 6s 141us/sample - loss: 1.7984 - acc: 0.3
Epoch 12/20
40000/40000 [==============================] - 6s 140us/sample - loss: 1.7790 - acc: 0.3
Epoch 13/20
40000/40000 [==============================] - 6s 138us/sample - loss: 1.7676 - acc: 0.4
Epoch 14/20
40000/40000 [==============================] - 6s 138us/sample - loss: 1.7609 - acc: 0.4
Epoch 15/20
40000/40000 [==============================] - 6s 138us/sample - loss: 1.7450 - acc: 0.4
Epoch 16/20
40000/40000 [==============================] - 6s 138us/sample - loss: 1.7496 - acc: 0.4
Epoch 17/20
40000/40000 [==============================] - 6s 141us/sample - loss: 1.7347 - acc: 0.4
Epoch 18/20
40000/40000 [==============================] - 6s 139us/sample - loss: 1.7330 - acc: 0.4
Epoch 19/20
40000/40000 [==============================] - 6s 138us/sample - loss: 1.7178 - acc: 0.4
Epoch 20/20
40000/40000 [==============================] - 6s 140us/sample - loss: 1.7105 - acc: 0.4
10000/10000 - 1s - loss: 1.7745 - acc: 0.3983

Test accuracy: 0.3983
dict_keys(['loss', 'acc', 'val_loss', 'val_acc'])
<matplotlib.legend.Legend at 0x7f7813816390>
```

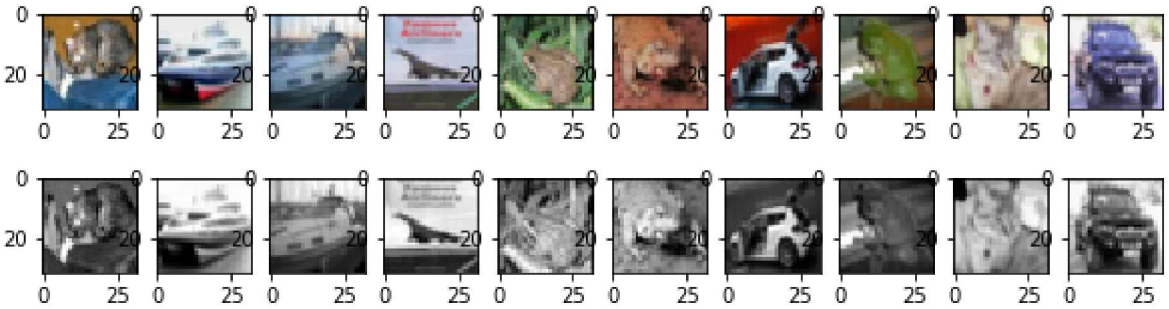




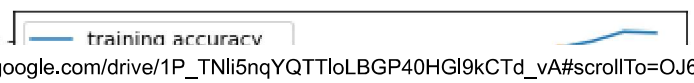
dropout

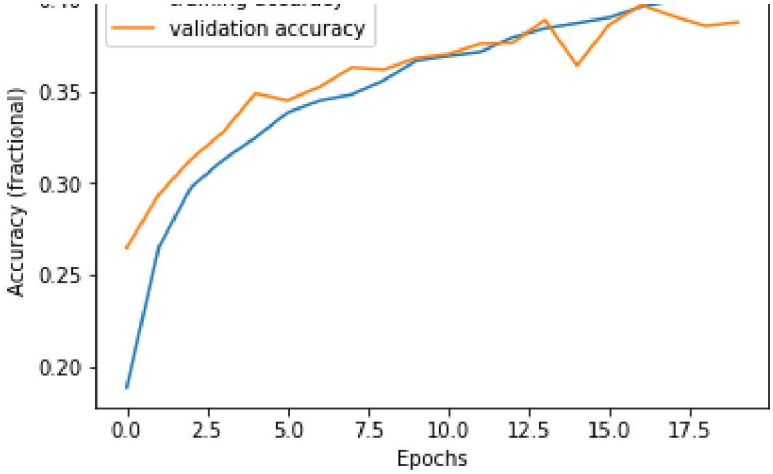
```
#implementing in Keras
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
from tensorflow import keras
from keras import regularizers

#(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()

#loading dataset
from keras.datasets import cifar10
(x_train, y_train), (x_test, y_test) = cifar10.load_data()

#1. Plotting the first 10 cifar10 images
# fig, axes = plt.subplots(1, 10, figsize=(10,5))
# for img, ax in zip(x_train[:10], axes):
#   ax.imshow(img)
# plt.show()

#2. Convert xtest and xtrain to float32 and normalize data to range 0.0 - 1.0
x_train_norm = np.float32(x_train/255.0)
x_test_norm = np.float32(x_test/255.0)
#print(x_train_norm)
#print(x_test_norm)

#3. function to convert to grayscale
def grayscale(data, dtype='float32'):
```

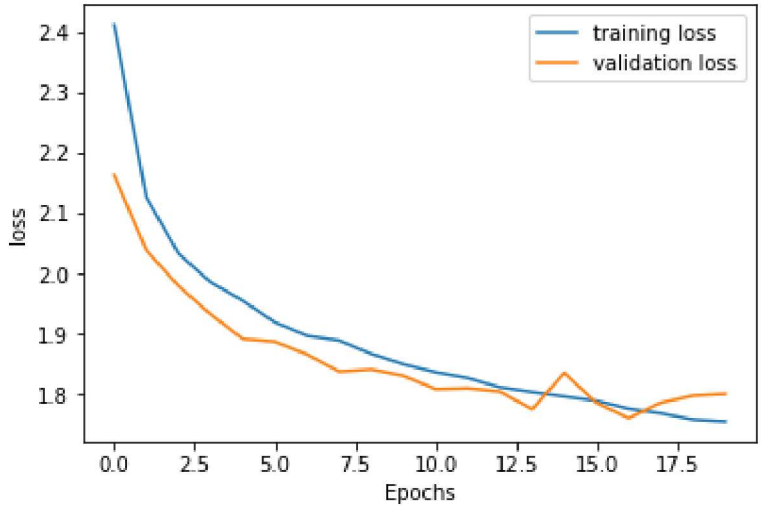
```
    # luma coding weighted average in video systems
    r, g, b = np.asarray(.3, dtype=dtype), np.asarray(.59, dtype=dtype), np.asarray(.11, dtype=dtype)
    rst = r * data[:, :, :, 0] + g * data[:, :, :, 1] + b * data[:, :, :, 2]
    # add channel dimension
    rst = np.expand_dims(rst, axis=3)
    return rst

#function Call
X_train_gray = grayscale(x_train_norm)
X_test_gray = grayscale(x_test_norm)


#plot a few grayscale vrs rgb
fig, axes = plt.subplots(1, 10, figsize=(10,5))
for img, ax in zip(x_test[:10], axes):
  ax.imshow(img)
plt.show()

fig, axes = plt.subplots(1, 10, figsize=(10,5))
for img, ax in zip(X_test_gray[:10], axes):
  ax.imshow(img[:,:,0], cmap=plt.get_cmap('gray'), interpolation='none')
plt.show()

#create ANN model
#model constants

#define network
l2_alpha = 0.0005
drop_rate = 0.2

model = keras.Sequential([
    keras.layers.Flatten(input_shape = (32, 32, 1)),
    keras.layers.Dense(512, activation = 'relu', kernel_regularizer=regularizers.l2(l2_alpha)),
    keras.layers.Dense(256, activation = 'relu', kernel_regularizer=regularizers.l2(l2_alpha)),
    keras.layers.Dense(64, activation = 'relu', kernel_regularizer=regularizers.l2(l2_alpha)),
    keras.layers.Dense(32, activation = 'relu', kernel_regularizer=regularizers.l2(l2_alpha)),
    keras.layers.Dropout(drop_rate, noise_shape=None, seed=None),
    keras.layers.Dense(10, activation = 'softmax'),
])

#Compile Network
model.compile(optimizer='adam', loss ='sparse_categorical_crossentropy',
              metrics = ['accuracy'])

#train the network with 20% validation
history = model.fit(X_train_gray, y_train, batch_size = 128, epochs=20, validation_split=0.2) #spits

#test accuracy and loss
test_loss, test_acc = model.evaluate(X_test_gray, y_test, verbose=2)
print('\nTest accuracy:', test_acc)

predictions = model.predict(X_test_gray)

predictions[0]

np.argmax(predictions[0])

print(history.history.keys())
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.xlabel('Epochs')
plt.ylabel('Accuracy (fractional)')
plt.legend(['training accuracy', 'validation accuracy'], loc='best')

plt.figure()
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.xlabel('Epochs')
plt.ylabel('loss')
plt.legend(['training loss', 'validation loss'], loc='best')
```



```
Train on 40000 samples, validate on 10000 samples
Epoch 1/20
40000/40000 [==============================] - 5s 137us/sample - loss: 2.4127 - acc: 0.1
Epoch 2/20
40000/40000 [==============================] - 5s 130us/sample - loss: 2.1266 - acc: 0.2
Epoch 3/20
40000/40000 [==============================] - 5s 131us/sample - loss: 2.0335 - acc: 0.2
Epoch 4/20
40000/40000 [==============================] - 5s 133us/sample - loss: 1.9858 - acc: 0.3
Epoch 5/20
40000/40000 [==============================] - 5s 130us/sample - loss: 1.9552 - acc: 0.3
Epoch 6/20
40000/40000 [==============================] - 5s 129us/sample - loss: 1.9191 - acc: 0.3
Epoch 7/20
40000/40000 [==============================] - 5s 132us/sample - loss: 1.8969 - acc: 0.3
Epoch 8/20
40000/40000 [==============================] - 5s 132us/sample - loss: 1.8890 - acc: 0.3
Epoch 9/20
40000/40000 [==============================] - 5s 130us/sample - loss: 1.8668 - acc: 0.3
Epoch 10/20
40000/40000 [==============================] - 5s 131us/sample - loss: 1.8497 - acc: 0.3
Epoch 11/20
40000/40000 [==============================] - 5s 132us/sample - loss: 1.8363 - acc: 0.3
Epoch 12/20
40000/40000 [==============================] - 5s 132us/sample - loss: 1.8269 - acc: 0.3
Epoch 13/20
40000/40000 [==============================] - 5s 132us/sample - loss: 1.8113 - acc: 0.3
Epoch 14/20
40000/40000 [==============================] - 5s 134us/sample - loss: 1.8034 - acc: 0.3
Epoch 15/20
40000/40000 [==============================] - 5s 133us/sample - loss: 1.7962 - acc: 0.3
Epoch 16/20
40000/40000 [==============================] - 5s 135us/sample - loss: 1.7890 - acc: 0.3
Epoch 17/20
40000/40000 [==============================] - 5s 132us/sample - loss: 1.7757 - acc: 0.3
Epoch 18/20
40000/40000 [==============================] - 5s 131us/sample - loss: 1.7683 - acc: 0.3
Epoch 19/20
40000/40000 [==============================] - 5s 130us/sample - loss: 1.7577 - acc: 0.4
Epoch 20/20
40000/40000 [==============================] - 5s 132us/sample - loss: 1.7544 - acc: 0.4
10000/10000 - 1s - loss: 1.8092 - acc: 0.3818

Test accuracy: 0.3818
dict_keys(['loss', 'acc', 'val_loss', 'val_acc'])
<matplotlib.legend.Legend at 0x7f6f6c99ccf8>
```




validation accuracy
Accuracy (fractional)
0.35
0.30
0.25
0.20
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Epochs


training loss
validation loss
loss
2.4
2.3
2.2
2.1
2.0
1.9
1.8
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Epochs