Classes of Signals

[1]:

```
class_labels = ['32PSK','16APSK','32QAM','FM','GMSK','32APSK','OQPSK','8ASK','BPSK','8PSK','AM-SSB-SC','4ASK','16PSK','64APSK','128QAM','128APSK','AM-D

'''
Index  Signal
0 --> '32PSK',
1 --> '16APSK',
2 --> '32QAM',
3 -->  'FM',
4 --> 'GMSK',
5 --> '32APSK',
6 --> 'OQPSK',
7 --> '8ASK',
8 --> 'BPSK',
9 --> '8PSK',
10 --> 'AM-SSB-SC',
11 --> '4ASK',
12 --> '16PSK',
13 --> '64APSK',
14 --> '128QAM',
15 --> '128APSK',
16 --> 'AM-DSB-SC',
17 --> 'AM-SSB-WC',
18 --> '64QAM',
19 --> 'QPSK',
20 --> '256QAM',
21 --> 'AM-DSB-WC',
22 --> 'OOK',
23 --> '16QAM']

'''
print(class_labels)
```

```
['32PSK', '16APSK', '32QAM', 'FM', 'GMSK', '32APSK', 'OQPSK', '8ASK', 'BPSK', '8PSK', 'AM-SSB-SC', '4ASK', '16PSK', '64APSK', '128QAM', '128APSK', 'AM
-DSB-SC', 'AM-SSB-WC', '64QAM', 'QPSK', '256QAM', 'AM-DSB-WC', 'OOK', '16QAM']
```

Importing Deep Learning Libraries

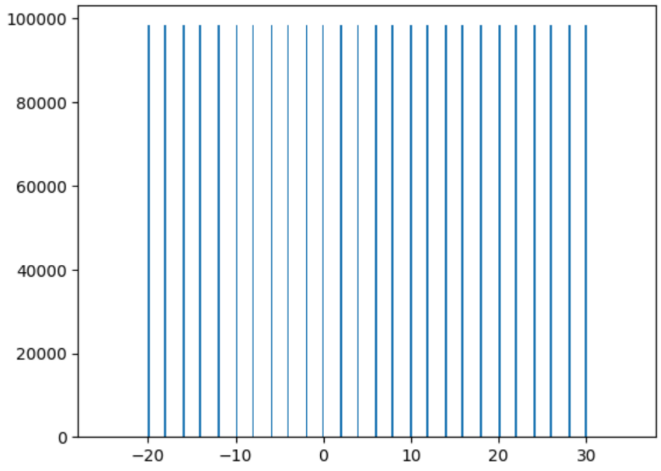
[2]:

```
import zipfile
# from google.colab import files
import os
import matplotlib.pyplot as plt
from matplotlib import pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
import numpy as np
import keras
from keras import layers
from tensorflow.keras.utils import to_categorical
from keras.models import Model, load_model
from keras.initializers import glorot_uniform
from keras.layers import Input, Dropout, Add, Dense, Reshape, Activation
from keras.layers import BatchNormalization, Flatten, Conv1D, MaxPooling1D
from tensorflow.keras.optimizers import Adam
```

Data Visualization

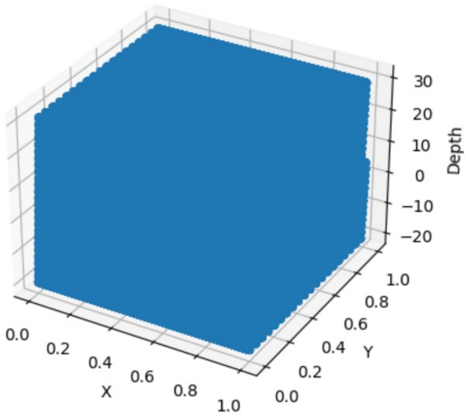
[3]:

```
data = np.load('gcs/snrs.npy')
plt.hist(data.ravel(),256,[-25,35])
plt.show()
```



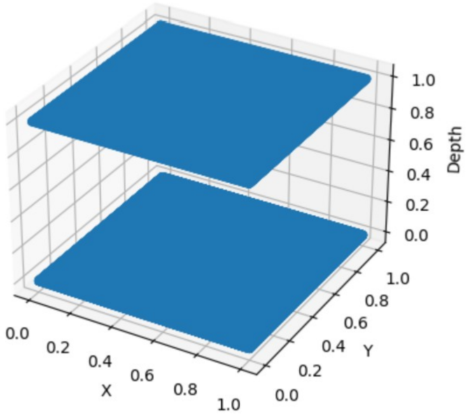
[4]:

```
img_array = np.load('gcs/snrs.npy')
img_array.resize(1080,1920)

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

x = np.linspace(0,1,1920)
y = np.linspace(0,1,1080)
z = img_array
x, y = np.meshgrid(x, y)
ax.scatter(x, y, z)

ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Depth');
```



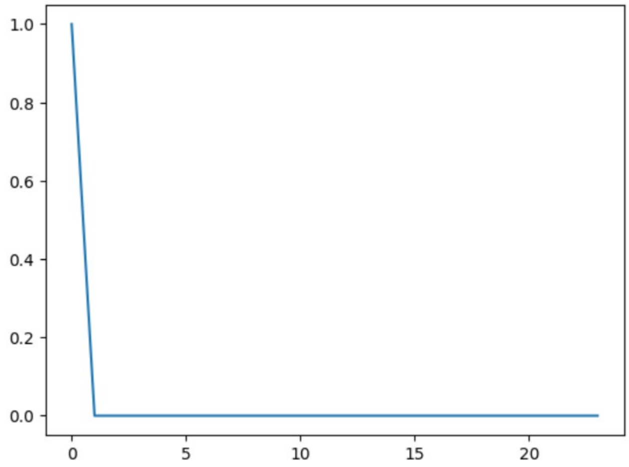
[5]:

```
img_array = np.load('gcs/labels.npy')
img_array.resize(1080,1920)

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

x = np.linspace(0,1,1920)
y = np.linspace(0,1,1080)
z = img_array
x, y = np.meshgrid(x, y)
ax.scatter(x, y, z)

ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Depth');
```



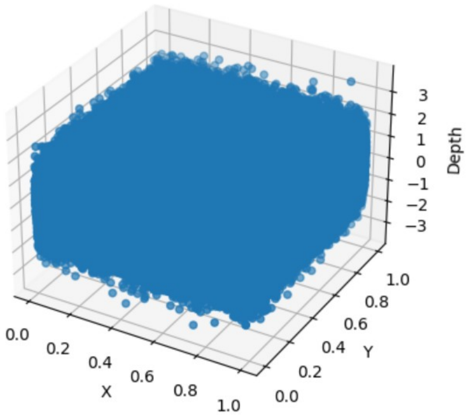
[7]:

```
data = np.load('gcs/labels.npy')
plt.plot(data[1000])
plt.show()
```



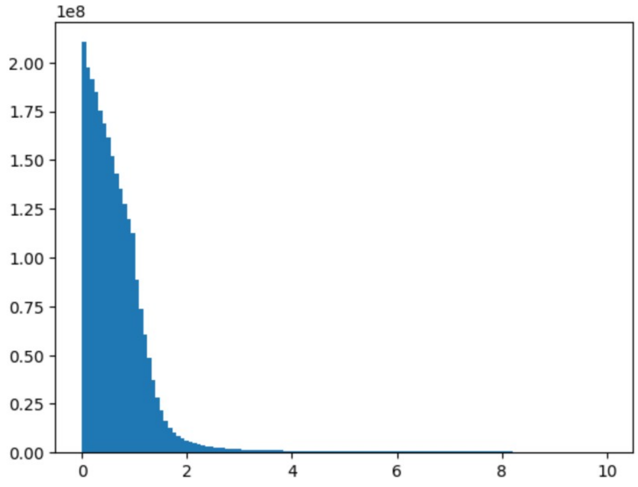
[8]:

```
img_array = np.load('gcs/signals.npy')
img_array.resize(1080,1920)

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

x = np.linspace(0,1,1920)
y = np.linspace(0,1,1080)
z = img_array
x, y = np.meshgrid(x, y)
ax.scatter(x, y, z)

ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Depth');
```



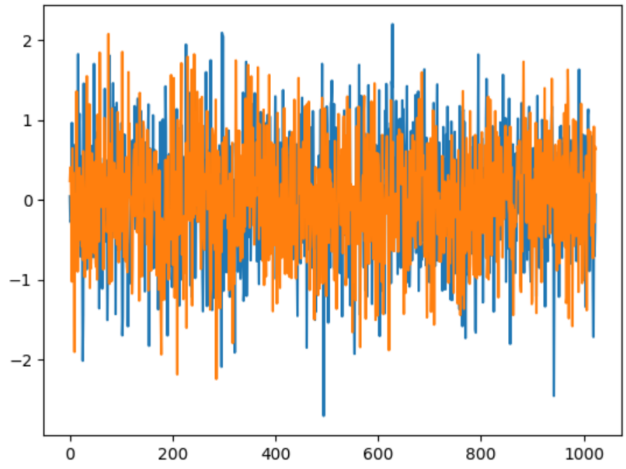
[9]:

```
data = np.load('gcs/signals.npy')
plt.hist(data.ravel(),128,[0,10])
plt.show()
```



[10]:

```
data = np.load('gcs/signals.npy')
plt.plot(data[0])
plt.show()
```



Getting and Loading the Data

[9]:
```
# Upload signals, labels, snrs
labels = np.load('gcs/labels.npy', mmap_mode = 'r')
signals = np.load('gcs/signals.npy', mmap_mode = 'r')
snrs = np.load('gcs/snrs.npy', mmap_mode = 'r')
```

[10]:
```
# Split arrays for two 2 parts (we take only second part of dataset of labeled signals because of the memory)
part = 2
signals = signals[::part, :, :] # 3D array
labels = labels[::part, :] # 2D array
snrs = snrs[::part, :] # 2D array

print(signals.shape)
print(labels.shape)
print(snrs.shape)
```

```
(1277952, 1024, 2)
(1277952, 24)
(1277952, 1)
```

[11]:
```
# Ndarray to array
snrs = np.ravel(snrs)
print(f"All possible SNRS: {np.unique(snrs)} db") # f string, return unique snrs
```

```
All possible SNRS: [-20. -18. -16. -14. -12. -10.  -8.  -6.  -4.  -2.   0.   2.   4.   6.
   8.  10.  12.  14.  16.  18.  20.  22.  24.  26.  28.  30.] db
```

[12]:
```
# Masked numpy array
c = np.ma.masked_where(snrs > 8, snrs)
print(c)
msk = c.mask
# Count unique elements in array
print(np.unique(c.mask, return_counts=True))
```

```
[-20.0 -20.0 -20.0 ... -- -- --]
(array([False,  True]), array([737280, 540672]))
```

[13]:
```
# Mask array of signals and labels (snrs > 8)
signals = signals[msk]
labels = labels[msk]

print(len(signals))
print(len(labels))
```

```
540672
540672
```

[14]:
```
# Train|test = 80|20

x_train,x_test, y_train, y_test = train_test_split(signals, labels, train_size=0.8, stratify=labels)

# print(f"Number of rows in y_train by class: {np.bincount(y_train)}")
# print(f"Number of rows in y_test by class: {np.bincount(y_test)}")
print(x_test.shape)
print(y_test.shape)

# Train|validation|test = 64|16|20
x_train, x_val, y_train, y_val = train_test_split(x_train, y_train, train_size=0.8, stratify=y_train)

# Validation is done for Model Hyperparameter Tuning, we share data from training to validation for this purpose

# print(f"Number of rows in y_train by class: {np.bincount(y_test)}")
# print(f"Number of rows in y_test by class: {np.bincount(y_val)}")
print(x_train.shape)
print(y_train.shape)
```

```
(108135, 1024, 2)
(108135, 24)
(346029, 1024, 2)
(346029, 24)
```

Creating Residual Block

[15]:

```
# Create residual and convolution block
class Residual_block:
    kernel_size = 3
    strides = 1
    padding = 'same'
    data_format = "channels_last"

    def __init__(self, x, x_shortcut, filters):
        self.x = x
        self.filters = filters
        self.x_shortcut = x_shortcut

    def unit(self):
        x = Conv1D(self.filters, self.kernel_size, self.strides, self.padding, self.data_format)(self.x)
        x = Activation('relu')(x)
        x = Conv1D(self.filters, self.kernel_size, self.strides, self.padding, self.data_format)(x)
        x = Activation('linear')(x)
        # add skip connection
        if x.shape[1:] == self.x_shortcut.shape[1:]:
            x = Add()([x, self.x_shortcut])
        else:
            raise Exception('Skip Connection Failure!')
        return x

class Convolution_block:
    kernel_size = 1
    strides = 1
    padding = 'same'
    data_format = "channels_last"

    def __init__(self, x, filters):
        self.x = x
        self.filters = filters

    def unit(self):
        x = Conv1D(self.filters, self.kernel_size, self.strides, self.padding, self.data_format)(self.x)
        x = Activation('linear')(x)
        return x
```

Creating Residual Stack

[16]:

```
# Create residual stack
def residual_stack(x, filters):
    x = Convolution_block(x, filters)
    print('x')
#     print(x.shape)
    print(x)
    x = x.unit()
    print('xunit')
#     print(x.shape)
    print(x)

    x_shortcut = x
    x = Residual_block(x, x_shortcut, filters)
    x = x.unit()
    x_shortcut = x
    x = Residual_block(x, x_shortcut, filters)
    x = x.unit()

    # max pooling layer
    x = MaxPooling1D(pool_size=2, strides=None, padding='valid', data_format='channels_last')(x)
#     print('Residual stack created')
    return x
```

Defining ResNet Model

[17]:

```
# define resnet model
def ResNet(input_shape, classes):
    # create input tensor
    x_input = Input(input_shape)
    x = x_input
    # residual stack
    num_filters = 40
    x = residual_stack(x, num_filters)
    x = residual_stack(x, num_filters)
    x = residual_stack(x, num_filters)
    x = residual_stack(x, num_filters)
    x = residual_stack(x, num_filters)

    # output layer
    x = Flatten()(x)
    x = Dense(128, activation="selu", kernel_initializer="he_normal")(x)
    x = Dropout(.5)(x)
    x = Dense(128, activation="selu", kernel_initializer="he_normal")(x)
    x = Dropout(.5)(x)
    x = Dense(classes , activation='softmax', kernel_initializer = glorot_uniform(seed=0))(x)

    # Create model
    model = Model(inputs = x_input, outputs = x)
#     print('Model ResNet created')
    return model
```

Save Model Weights and History

[18]:

```python
# option to save model weights and model history
save_model = True
save_history = True

# create directory for model weights
if save_model is True:
    weights_path = input("Name model weights directory: ")
    weights_path = "data/" + weights_path

    try:
        os.mkdir(weights_path)
    except OSError:
        print ("Creation of the directory %s failed" % weights_path)
    else:
        print ("Successfully created the directory %s " % weights_path)
    print('\n')


# create directory for model history
if save_history is True:
    history_path = input("Name model history directory: ")
    history_path = "data/" + history_path

    try:
        os.mkdir(history_path)
    except OSError:
        print ("Creation of the directory %s failed" % history_path)
    else:
        print ("Successfully created the directory %s " % history_path)
    print('\n')
```

```
Name model weights directory:  wts
Successfully created the directory data/wts


Name model history directory:  hist
Successfully created the directory data/hist
```

Set Model Parameters

[19]:

```python
# initialize optimizer
adm = Adam(learning_rate=0.0001, beta_1=0.9, beta_2=0.999, epsilon=None, decay=0.0, amsgrad=False)

# set number of epochs
num_epochs = int(input('Enter number of epochs: '))

# set batch size
batch = 32

# configure weights save

if save_model is True:
    filepath= weights_path + "/{epoch}.hdf5"
    checkpoint = keras.callbacks.ModelCheckpoint(filepath, monitor='val_acc', verbose=1, save_best_only=False, mode="auto")
    callbacks_list = [checkpoint]
else:
    callbacks_list = []
```

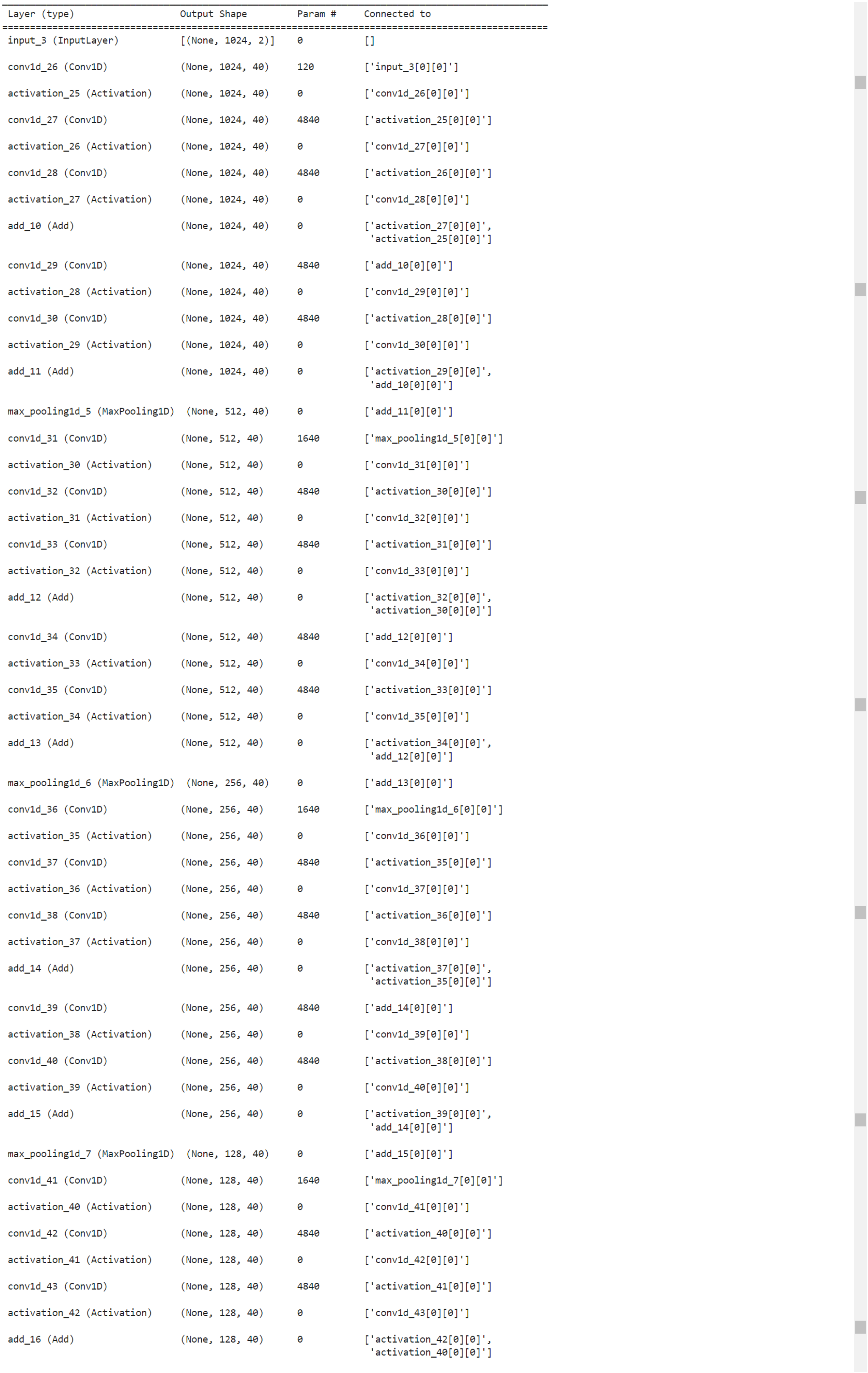
```
Enter number of epochs:  40
```

Print Model Summary and Train Network

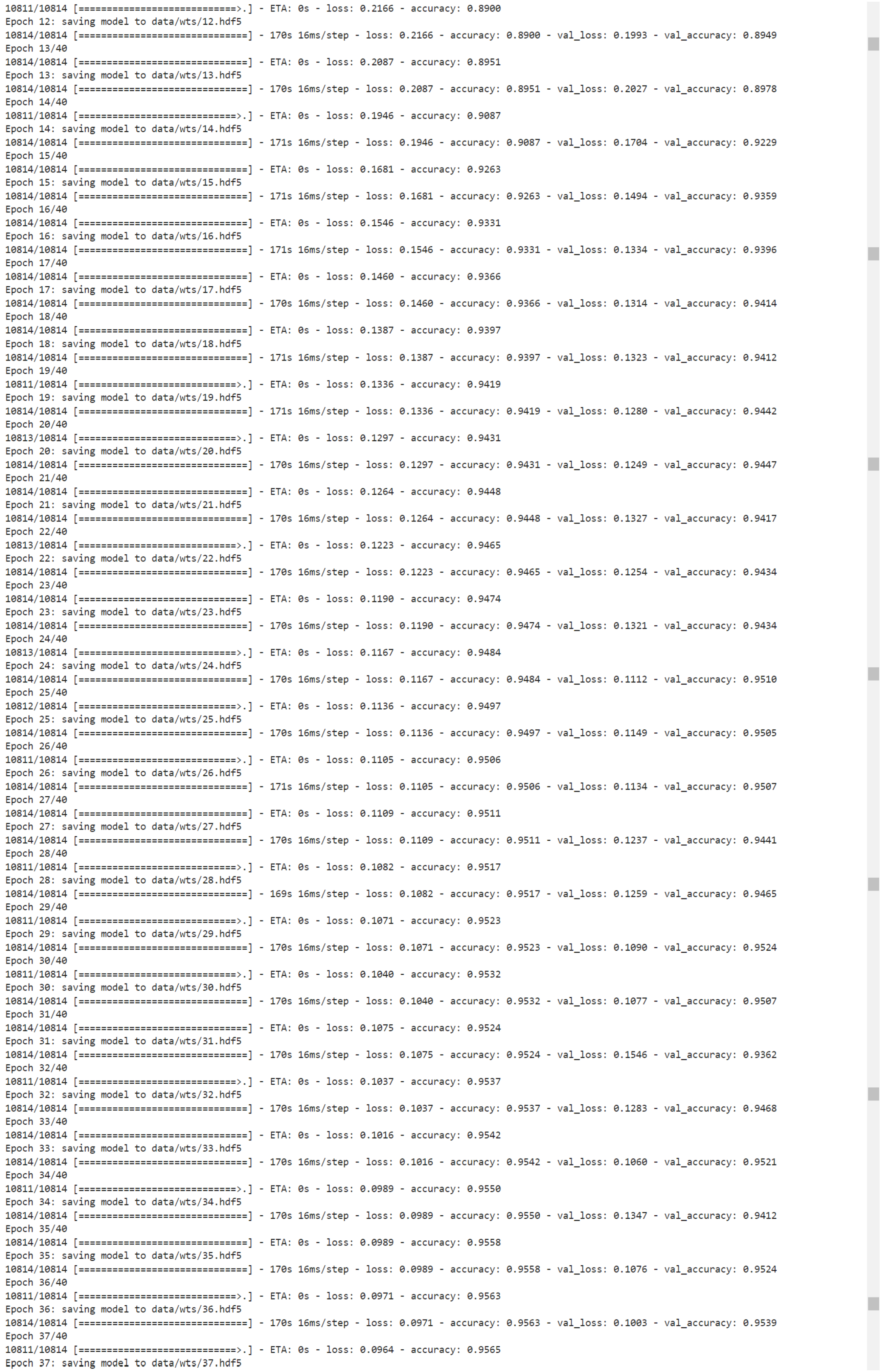
[22]:

```python
# initialize and train model
model = ResNet((1024, 2), 24)
model.compile(optimizer=adm, loss='categorical_crossentropy', metrics=['accuracy'])
model.summary()
history = model.fit(x_train, y_train, epochs = num_epochs, batch_size = batch, callbacks=callbacks_list, validation_data=(x_val, y_val))
```

```
x
<__main__.Convolution_block object at 0x7f82045a3bd0>
xunit
KerasTensor(type_spec=TensorSpec(shape=(None, 1024, 40), dtype=tf.float32, name=None), name='Placeholder:0', description="created by layer 'activation
_25'")
x
<__main__.Convolution_block object at 0x7f81f878ea90>
xunit
KerasTensor(type_spec=TensorSpec(shape=(None, 512, 40), dtype=tf.float32, name=None), name='Placeholder:0', description="created by layer 'activation_
30'")
x
<__main__.Convolution_block object at 0x7f81f878ea90>
xunit
KerasTensor(type_spec=TensorSpec(shape=(None, 256, 40), dtype=tf.float32, name=None), name='Placeholder:0', description="created by layer 'activation_
35'")
x
<__main__.Convolution_block object at 0x7f81f87cbb10>
xunit
KerasTensor(type_spec=TensorSpec(shape=(None, 128, 40), dtype=tf.float32, name=None), name='Placeholder:0', description="created by layer 'activation_
40'")
x
<__main__.Convolution_block object at 0x7f81f878bb10>
xunit
KerasTensor(type_spec=TensorSpec(shape=(None, 64, 40), dtype=tf.float32, name=None), name='Placeholder:0', description="created by layer 'activation_4
5'")
Model: "model_1"
```



```
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_3 (InputLayer)           [(None, 1024, 2)]    0           []

 conv1d_26 (Conv1D)             (None, 1024, 40)     120         ['input_3[0][0]']

 activation_25 (Activation)     (None, 1024, 40)     0           ['conv1d_26[0][0]']

 conv1d_27 (Conv1D)             (None, 1024, 40)     4840        ['activation_25[0][0]']

 activation_26 (Activation)     (None, 1024, 40)     0           ['conv1d_27[0][0]']

 conv1d_28 (Conv1D)             (None, 1024, 40)     4840        ['activation_26[0][0]']

 activation_27 (Activation)     (None, 1024, 40)     0           ['conv1d_28[0][0]']

 add_10 (Add)                   (None, 1024, 40)     0           ['activation_27[0][0]',
                                                                  'activation_25[0][0]']

 conv1d_29 (Conv1D)             (None, 1024, 40)     4840        ['add_10[0][0]']

 activation_28 (Activation)     (None, 1024, 40)     0           ['conv1d_29[0][0]']

 conv1d_30 (Conv1D)             (None, 1024, 40)     4840        ['activation_28[0][0]']

 activation_29 (Activation)     (None, 1024, 40)     0           ['conv1d_30[0][0]']

 add_11 (Add)                   (None, 1024, 40)     0           ['activation_29[0][0]',
                                                                  'add_10[0][0]']

 max_pooling1d_5 (MaxPooling1D)  (None, 512, 40)     0           ['add_11[0][0]']

 conv1d_31 (Conv1D)             (None, 512, 40)      1640        ['max_pooling1d_5[0][0]']

 activation_30 (Activation)     (None, 512, 40)      0           ['conv1d_31[0][0]']

 conv1d_32 (Conv1D)             (None, 512, 40)      4840        ['activation_30[0][0]']

 activation_31 (Activation)     (None, 512, 40)      0           ['conv1d_32[0][0]']

 conv1d_33 (Conv1D)             (None, 512, 40)      4840        ['activation_31[0][0]']

 activation_32 (Activation)     (None, 512, 40)      0           ['conv1d_33[0][0]']

 add_12 (Add)                   (None, 512, 40)      0           ['activation_32[0][0]',
                                                                  'activation_30[0][0]']

 conv1d_34 (Conv1D)             (None, 512, 40)      4840        ['add_12[0][0]']

 activation_33 (Activation)     (None, 512, 40)      0           ['conv1d_34[0][0]']

 conv1d_35 (Conv1D)             (None, 512, 40)      4840        ['activation_33[0][0]']

 activation_34 (Activation)     (None, 512, 40)      0           ['conv1d_35[0][0]']

 add_13 (Add)                   (None, 512, 40)      0           ['activation_34[0][0]',
                                                                  'add_12[0][0]']

 max_pooling1d_6 (MaxPooling1D)  (None, 256, 40)     0           ['add_13[0][0]']

 conv1d_36 (Conv1D)             (None, 256, 40)      1640        ['max_pooling1d_6[0][0]']

 activation_35 (Activation)     (None, 256, 40)      0           ['conv1d_36[0][0]']

 conv1d_37 (Conv1D)             (None, 256, 40)      4840        ['activation_35[0][0]']

 activation_36 (Activation)     (None, 256, 40)      0           ['conv1d_37[0][0]']

 conv1d_38 (Conv1D)             (None, 256, 40)      4840        ['activation_36[0][0]']

 activation_37 (Activation)     (None, 256, 40)      0           ['conv1d_38[0][0]']

 add_14 (Add)                   (None, 256, 40)      0           ['activation_37[0][0]',
                                                                  'activation_35[0][0]']

 conv1d_39 (Conv1D)             (None, 256, 40)      4840        ['add_14[0][0]']

 activation_38 (Activation)     (None, 256, 40)      0           ['conv1d_39[0][0]']

 conv1d_40 (Conv1D)             (None, 256, 40)      4840        ['activation_38[0][0]']

 activation_39 (Activation)     (None, 256, 40)      0           ['conv1d_40[0][0]']

 add_15 (Add)                   (None, 256, 40)      0           ['activation_39[0][0]',
                                                                  'add_14[0][0]']

 max_pooling1d_7 (MaxPooling1D)  (None, 128, 40)     0           ['add_15[0][0]']

 conv1d_41 (Conv1D)             (None, 128, 40)      1640        ['max_pooling1d_7[0][0]']

 activation_40 (Activation)     (None, 128, 40)      0           ['conv1d_41[0][0]']

 conv1d_42 (Conv1D)             (None, 128, 40)      4840        ['activation_40[0][0]']

 activation_41 (Activation)     (None, 128, 40)      0           ['conv1d_42[0][0]']

 conv1d_43 (Conv1D)             (None, 128, 40)      4840        ['activation_41[0][0]']

 activation_42 (Activation)     (None, 128, 40)      0           ['conv1d_43[0][0]']

 add_16 (Add)                   (None, 128, 40)      0           ['activation_42[0][0]',
                                                                  'activation_40[0][0]']
 conv1d_44 (Conv1D)             (None, 128, 40)      4840        ['add_16[0][0]']

 activation_43 (Activation)     (None, 128, 40)      0           ['conv1d_44[0][0]']

 conv1d_45 (Conv1D)             (None, 128, 40)      4840        ['activation_43[0][0]']

 activation_44 (Activation)     (None, 128, 40)      0           ['conv1d_45[0][0]']

 add_17 (Add)                   (None, 128, 40)      0           ['activation_44[0][0]',
                                                                  'add_16[0][0]']

 max_pooling1d_8 (MaxPooling1D)  (None, 64, 40)      0           ['add_17[0][0]']

 conv1d_46 (Conv1D)             (None, 64, 40)       1640        ['max_pooling1d_8[0][0]']

 activation_45 (Activation)     (None, 64, 40)       0           ['conv1d_46[0][0]']

 conv1d_47 (Conv1D)             (None, 64, 40)       4840        ['activation_45[0][0]']

 activation_46 (Activation)     (None, 64, 40)       0           ['conv1d_47[0][0]']

 conv1d_48 (Conv1D)             (None, 64, 40)       4840        ['activation_46[0][0]']

 activation_47 (Activation)     (None, 64, 40)       0           ['conv1d_48[0][0]']

 add_18 (Add)                   (None, 64, 40)       0           ['activation_47[0][0]',
                                                                  'activation_45[0][0]']

 conv1d_49 (Conv1D)             (None, 64, 40)       4840        ['add_18[0][0]']

 activation_48 (Activation)     (None, 64, 40)       0           ['conv1d_49[0][0]']

 conv1d_50 (Conv1D)             (None, 64, 40)       4840        ['activation_48[0][0]']

 activation_49 (Activation)     (None, 64, 40)       0           ['conv1d_50[0][0]']

 add_19 (Add)                   (None, 64, 40)       0           ['activation_49[0][0]',
                                                                  'add_18[0][0]']

 max_pooling1d_9 (MaxPooling1D)  (None, 32, 40)      0           ['add_19[0][0]']

 flatten_1 (Flatten)            (None, 1280)         0           ['max_pooling1d_9[0][0]']

 dense_3 (Dense)                (None, 128)          163968      ['flatten_1[0][0]']

 dropout_2 (Dropout)            (None, 128)          0           ['dense_3[0][0]']

 dense_4 (Dense)                (None, 128)          16512       ['dropout_2[0][0]']

 dropout_3 (Dropout)            (None, 128)          0           ['dense_4[0][0]']

 dense_5 (Dense)                (None, 24)           3096        ['dropout_3[0][0]']

==================================================================================================
Total params: 287,056
Trainable params: 287,056
Non-trainable params: 0
__________________________________________________________________________________________________
Epoch 1/40
10814/10814 [==============================] - ETA: 0s - loss: 1.6553 - accuracy: 0.4163
Epoch 1: saving model to data/wts/1.hdf5
10814/10814 [==============================] - 174s 16ms/step - loss: 1.6553 - accuracy: 0.4163 - val_loss: 0.9566 - val_accuracy: 0.5843
Epoch 2/40
10813/10814 [============================>.] - ETA: 0s - loss: 0.7808 - accuracy: 0.6557
Epoch 2: saving model to data/wts/2.hdf5
10814/10814 [==============================] - 171s 16ms/step - loss: 0.7808 - accuracy: 0.6557 - val_loss: 0.5561 - val_accuracy: 0.7363
Epoch 3/40
10814/10814 [==============================] - ETA: 0s - loss: 0.5394 - accuracy: 0.7449
Epoch 3: saving model to data/wts/3.hdf5
10814/10814 [==============================] - 172s 16ms/step - loss: 0.5394 - accuracy: 0.7449 - val_loss: 0.4247 - val_accuracy: 0.7870
Epoch 4/40
10811/10814 [============================>.] - ETA: 0s - loss: 0.4440 - accuracy: 0.7832
Epoch 4: saving model to data/wts/4.hdf5
10814/10814 [==============================] - 171s 16ms/step - loss: 0.4439 - accuracy: 0.7832 - val_loss: 0.4090 - val_accuracy: 0.7963
Epoch 5/40
10812/10814 [============================>.] - ETA: 0s - loss: 0.3954 - accuracy: 0.8039
Epoch 5: saving model to data/wts/5.hdf5
10814/10814 [==============================] - 170s 16ms/step - loss: 0.3954 - accuracy: 0.8039 - val_loss: 0.4237 - val_accuracy: 0.7972
Epoch 6/40
10812/10814 [============================>.] - ETA: 0s - loss: 0.3663 - accuracy: 0.8176
Epoch 6: saving model to data/wts/6.hdf5
10814/10814 [==============================] - 170s 16ms/step - loss: 0.3663 - accuracy: 0.8176 - val_loss: 0.3599 - val_accuracy: 0.8184
Epoch 7/40
10812/10814 [============================>.] - ETA: 0s - loss: 0.3362 - accuracy: 0.8363
Epoch 7: saving model to data/wts/7.hdf5
10814/10814 [==============================] - 170s 16ms/step - loss: 0.3362 - accuracy: 0.8364 - val_loss: 0.2639 - val_accuracy: 0.8714
Epoch 8/40
10813/10814 [============================>.] - ETA: 0s - loss: 0.2728 - accuracy: 0.8663
Epoch 8: saving model to data/wts/8.hdf5
10814/10814 [==============================] - 171s 16ms/step - loss: 0.2728 - accuracy: 0.8663 - val_loss: 0.2267 - val_accuracy: 0.8837
Epoch 9/40
10814/10814 [==============================] - ETA: 0s - loss: 0.2479 - accuracy: 0.8759
Epoch 9: saving model to data/wts/9.hdf5
10814/10814 [==============================] - 171s 16ms/step - loss: 0.2479 - accuracy: 0.8759 - val_loss: 0.2406 - val_accuracy: 0.8799
Epoch 10/40
10814/10814 [==============================] - ETA: 0s - loss: 0.2361 - accuracy: 0.8813
Epoch 10: saving model to data/wts/10.hdf5
10814/10814 [==============================] - 171s 16ms/step - loss: 0.2361 - accuracy: 0.8813 - val_loss: 0.2079 - val_accuracy: 0.8920
Epoch 11/40
10814/10814 [==============================] - ETA: 0s - loss: 0.2248 - accuracy: 0.8856
Epoch 11: saving model to data/wts/11.hdf5
10814/10814 [==============================] - 170s 16ms/step - loss: 0.2248 - accuracy: 0.8856 - val_loss: 0.2146 - val_accuracy: 0.8898
Epoch 12/40
```



```
10811/10814 [============================>.] - ETA: 0s - loss: 0.2166 - accuracy: 0.8900
Epoch 12: saving model to data/wts/12.hdf5
10814/10814 [==============================] - 170s 16ms/step - loss: 0.2166 - accuracy: 0.8900 - val_loss: 0.1993 - val_accuracy: 0.8949
Epoch 13/40
10814/10814 [==============================] - ETA: 0s - loss: 0.2087 - accuracy: 0.8951
Epoch 13: saving model to data/wts/13.hdf5
10814/10814 [==============================] - 170s 16ms/step - loss: 0.2087 - accuracy: 0.8951 - val_loss: 0.2027 - val_accuracy: 0.8978
Epoch 14/40
10811/10814 [============================>.] - ETA: 0s - loss: 0.1946 - accuracy: 0.9087
Epoch 14: saving model to data/wts/14.hdf5
10814/10814 [==============================] - 171s 16ms/step - loss: 0.1946 - accuracy: 0.9087 - val_loss: 0.1704 - val_accuracy: 0.9229
Epoch 15/40
10814/10814 [==============================] - ETA: 0s - loss: 0.1681 - accuracy: 0.9263
Epoch 15: saving model to data/wts/15.hdf5
10814/10814 [==============================] - 171s 16ms/step - loss: 0.1681 - accuracy: 0.9263 - val_loss: 0.1494 - val_accuracy: 0.9359
Epoch 16/40
10814/10814 [==============================] - ETA: 0s - loss: 0.1546 - accuracy: 0.9331
Epoch 16: saving model to data/wts/16.hdf5
10814/10814 [==============================] - 171s 16ms/step - loss: 0.1546 - accuracy: 0.9331 - val_loss: 0.1334 - val_accuracy: 0.9396
Epoch 17/40
10814/10814 [==============================] - ETA: 0s - loss: 0.1460 - accuracy: 0.9366
Epoch 17: saving model to data/wts/17.hdf5
10814/10814 [==============================] - 170s 16ms/step - loss: 0.1460 - accuracy: 0.9366 - val_loss: 0.1314 - val_accuracy: 0.9414
Epoch 18/40
10814/10814 [==============================] - ETA: 0s - loss: 0.1387 - accuracy: 0.9397
Epoch 18: saving model to data/wts/18.hdf5
10814/10814 [==============================] - 171s 16ms/step - loss: 0.1387 - accuracy: 0.9397 - val_loss: 0.1323 - val_accuracy: 0.9412
Epoch 19/40
10811/10814 [============================>.] - ETA: 0s - loss: 0.1336 - accuracy: 0.9419
Epoch 19: saving model to data/wts/19.hdf5
10814/10814 [==============================] - 171s 16ms/step - loss: 0.1336 - accuracy: 0.9419 - val_loss: 0.1280 - val_accuracy: 0.9442
Epoch 20/40
10813/10814 [============================>.] - ETA: 0s - loss: 0.1297 - accuracy: 0.9431
Epoch 20: saving model to data/wts/20.hdf5
10814/10814 [==============================] - 170s 16ms/step - loss: 0.1297 - accuracy: 0.9431 - val_loss: 0.1249 - val_accuracy: 0.9447
Epoch 21/40
10814/10814 [==============================] - ETA: 0s - loss: 0.1264 - accuracy: 0.9448
Epoch 21: saving model to data/wts/21.hdf5
10814/10814 [==============================] - 170s 16ms/step - loss: 0.1264 - accuracy: 0.9448 - val_loss: 0.1327 - val_accuracy: 0.9417
Epoch 22/40
10813/10814 [============================>.] - ETA: 0s - loss: 0.1223 - accuracy: 0.9465
Epoch 22: saving model to data/wts/22.hdf5
10814/10814 [==============================] - 170s 16ms/step - loss: 0.1223 - accuracy: 0.9465 - val_loss: 0.1254 - val_accuracy: 0.9434
Epoch 23/40
10814/10814 [==============================] - ETA: 0s - loss: 0.1190 - accuracy: 0.9474
Epoch 23: saving model to data/wts/23.hdf5
10814/10814 [==============================] - 170s 16ms/step - loss: 0.1190 - accuracy: 0.9474 - val_loss: 0.1321 - val_accuracy: 0.9434
Epoch 24/40
10813/10814 [============================>.] - ETA: 0s - loss: 0.1167 - accuracy: 0.9484
Epoch 24: saving model to data/wts/24.hdf5
10814/10814 [==============================] - 170s 16ms/step - loss: 0.1167 - accuracy: 0.9484 - val_loss: 0.1112 - val_accuracy: 0.9510
Epoch 25/40
10812/10814 [============================>.] - ETA: 0s - loss: 0.1136 - accuracy: 0.9497
Epoch 25: saving model to data/wts/25.hdf5
10814/10814 [==============================] - 170s 16ms/step - loss: 0.1136 - accuracy: 0.9497 - val_loss: 0.1149 - val_accuracy: 0.9505
Epoch 26/40
10811/10814 [============================>.] - ETA: 0s - loss: 0.1105 - accuracy: 0.9506
Epoch 26: saving model to data/wts/26.hdf5
10814/10814 [==============================] - 171s 16ms/step - loss: 0.1105 - accuracy: 0.9506 - val_loss: 0.1134 - val_accuracy: 0.9507
Epoch 27/40
10814/10814 [==============================] - ETA: 0s - loss: 0.1109 - accuracy: 0.9511
Epoch 27: saving model to data/wts/27.hdf5
10814/10814 [==============================] - 170s 16ms/step - loss: 0.1109 - accuracy: 0.9511 - val_loss: 0.1237 - val_accuracy: 0.9441
Epoch 28/40
10811/10814 [============================>.] - ETA: 0s - loss: 0.1082 - accuracy: 0.9517
Epoch 28: saving model to data/wts/28.hdf5
10814/10814 [==============================] - 169s 16ms/step - loss: 0.1082 - accuracy: 0.9517 - val_loss: 0.1259 - val_accuracy: 0.9465
Epoch 29/40
10811/10814 [============================>.] - ETA: 0s - loss: 0.1071 - accuracy: 0.9523
Epoch 29: saving model to data/wts/29.hdf5
10814/10814 [==============================] - 170s 16ms/step - loss: 0.1071 - accuracy: 0.9523 - val_loss: 0.1090 - val_accuracy: 0.9524
Epoch 30/40
10811/10814 [============================>.] - ETA: 0s - loss: 0.1040 - accuracy: 0.9532
Epoch 30: saving model to data/wts/30.hdf5
10814/10814 [==============================] - 170s 16ms/step - loss: 0.1040 - accuracy: 0.9532 - val_loss: 0.1077 - val_accuracy: 0.9507
Epoch 31/40
10814/10814 [==============================] - ETA: 0s - loss: 0.1075 - accuracy: 0.9524
Epoch 31: saving model to data/wts/31.hdf5
10814/10814 [==============================] - 170s 16ms/step - loss: 0.1075 - accuracy: 0.9524 - val_loss: 0.1546 - val_accuracy: 0.9362
Epoch 32/40
10811/10814 [============================>.] - ETA: 0s - loss: 0.1037 - accuracy: 0.9537
Epoch 32: saving model to data/wts/32.hdf5
10814/10814 [==============================] - 170s 16ms/step - loss: 0.1037 - accuracy: 0.9537 - val_loss: 0.1283 - val_accuracy: 0.9468
Epoch 33/40
10814/10814 [==============================] - ETA: 0s - loss: 0.1016 - accuracy: 0.9542
Epoch 33: saving model to data/wts/33.hdf5
10814/10814 [==============================] - 170s 16ms/step - loss: 0.1016 - accuracy: 0.9542 - val_loss: 0.1060 - val_accuracy: 0.9521
Epoch 34/40
10811/10814 [============================>.] - ETA: 0s - loss: 0.0989 - accuracy: 0.9550
Epoch 34: saving model to data/wts/34.hdf5
10814/10814 [==============================] - 170s 16ms/step - loss: 0.0989 - accuracy: 0.9550 - val_loss: 0.1347 - val_accuracy: 0.9412
Epoch 35/40
10814/10814 [==============================] - ETA: 0s - loss: 0.0989 - accuracy: 0.9558
Epoch 35: saving model to data/wts/35.hdf5
10814/10814 [==============================] - 170s 16ms/step - loss: 0.0989 - accuracy: 0.9558 - val_loss: 0.1076 - val_accuracy: 0.9524
Epoch 36/40
10811/10814 [============================>.] - ETA: 0s - loss: 0.0971 - accuracy: 0.9563
Epoch 36: saving model to data/wts/36.hdf5
10814/10814 [==============================] - 170s 16ms/step - loss: 0.0971 - accuracy: 0.9563 - val_loss: 0.1003 - val_accuracy: 0.9539
Epoch 37/40
10811/10814 [============================>.] - ETA: 0s - loss: 0.0964 - accuracy: 0.9565
Epoch 37: saving model to data/wts/37.hdf5
```

```
10814/10814 [==============================] - 170s 16ms/step - loss: 0.0964 - accuracy: 0.9565 - val_loss: 0.1454 - val_accuracy: 0.9399
Epoch 38/40
10813/10814 [============================>.] - ETA: 0s - loss: 0.0962 - accuracy: 0.9566
Epoch 38: saving model to data/wts/38.hdf5
10814/10814 [==============================] - 170s 16ms/step - loss: 0.0962 - accuracy: 0.9566 - val_loss: 0.1006 - val_accuracy: 0.9562
Epoch 39/40
10813/10814 [============================>.] - ETA: 0s - loss: 0.0948 - accuracy: 0.9574
Epoch 39: saving model to data/wts/39.hdf5
10814/10814 [==============================] - 170s 16ms/step - loss: 0.0948 - accuracy: 0.9574 - val_loss: 0.0984 - val_accuracy: 0.9570
Epoch 40/40
10813/10814 [============================>.] - ETA: 0s - loss: 0.0945 - accuracy: 0.9578
Epoch 40: saving model to data/wts/40.hdf5
10814/10814 [==============================] - 170s 16ms/step - loss: 0.0945 - accuracy: 0.9578 - val_loss: 0.0951 - val_accuracy: 0.9589
```

Save Model History

[23]:

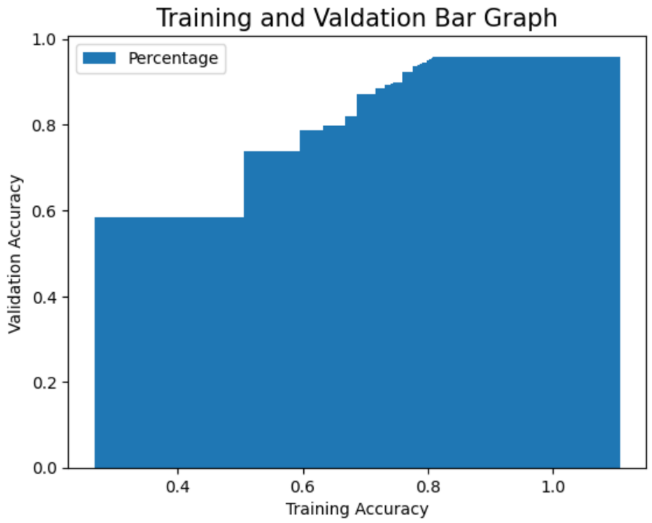
```
# record model history
train_accuracy = history.history['accuracy']
train_loss = history.history['loss']
val_accuracy = history.history['val_accuracy']
val_loss = history.history['val_loss']

if save_history is True:
    # save model history: loss and accuracy
    np.save(history_path + 'train_acc.npy', train_accuracy)
    np.save(history_path + 'train_loss.npy', train_loss)
    np.save(history_path + 'val_acc.npy', val_accuracy)
    np.save(history_path + 'val_loss.npy', val_loss)
    print("Model History Saved!")
    print('\n')
```

```
Model History Saved!
```

Model Training Accuracy

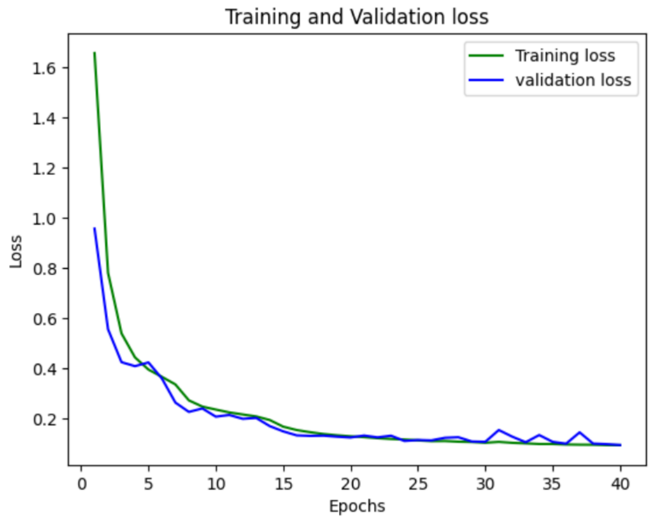
[24]:

```
x = train_accuracy
y= val_accuracy

plt.xlabel("Training Accuracy",fontsize=10)
plt.ylabel("Validation Accuracy",fontsize=10)
plt.title("Training and Valdation Bar Graph",fontsize=15)

plt.bar(x,y,width=0.3,align='center',label='Percentage')
plt.legend()

plt.show()
```

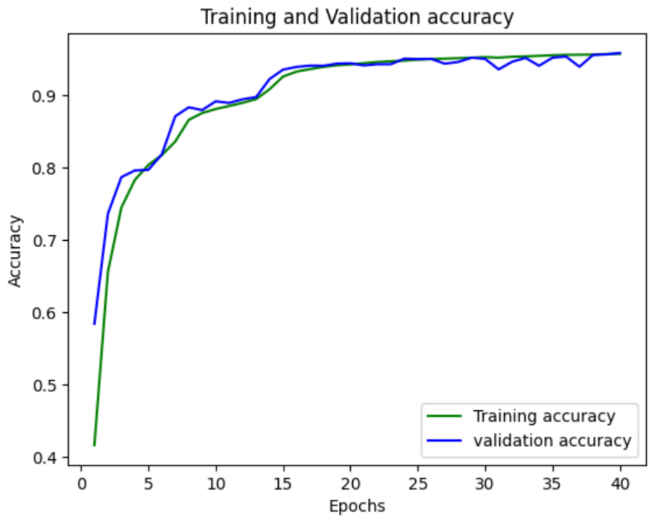


[26]:

```
loss_train = train_loss
loss_val = val_loss
epochs = range(1,41)
plt.plot(epochs, loss_train, 'g', label='Training loss')
plt.plot(epochs, loss_val, 'b', label='validation loss')
plt.title('Training and Validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```



```
[27]: acc_train = train_accuracy
      acc_val = val_accuracy
      epochs = range(1,41)
      plt.plot(epochs, acc_train, 'g', label='Training accuracy')
      plt.plot(epochs, acc_val, 'b', label='validation accuracy')
      plt.title('Training and Validation accuracy')
      plt.xlabel('Epochs')
      plt.ylabel('Accuracy')
      plt.legend()
      plt.show()
```



Evaluating Model Performace: on Test Data

```
[28]: model.predict(np.expand_dims(x_test[0], axis=0)).round(2)
```

```
[28]: array([[0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
              0., 0., 0., 0., 0., 0., 0., 0.]], dtype=float32)
```

Input Signal: Actual

```
[35]: # 11th class is the actial value of 4th signal in dataset
      np.argmax(y_test[108134])
```

```
[35]: 5
```

Output Signal: Prediction

```
[36]: # 11th class value predicted by model for 4th signal in dataset
      np.argmax(model.predict(np.expand_dims(x_test[108134], axis=0)).round(2))
```

```
[36]: 5
```

```
[31]: y_pred = model.predict(x_test).round(2)
```

```
[32]: # evaluate model on test data
      loss, acc = model.evaluate(x_test, y_test, batch_size=32)
      print('EVALUATING MODEL ON TEST DATA:')
      print('Test Accuracy: ', str(round(acc*100, 2)), '%') # calculated by compairing real o/p with the model prediction
      print('\n')
```

```
3380/3380 [==============================] - 17s 5ms/step - loss: 0.0953 - accuracy: 0.9577
EVALUATING MODEL ON TEST DATA:
Test Accuracy:  95.77 %
```

Classification Report

```
[38]: from sklearn.metrics import classification_report
      y_pred = (y_pred > 0)
      print(classification_report(y_test, y_pred, target_names= class_labels,output_dict=False, zero_division='warn'))
```

```
              precision    recall  f1-score   support

       32PSK       1.00      1.00      1.00      4506
      16APSK       1.00      1.00      1.00      4506
       32QAM       1.00      1.00      1.00      4506
          FM       1.00      1.00      1.00      4506
        GMSK       1.00      1.00      1.00      4505
      32APSK       1.00      1.00      1.00      4506
       OQPSK       0.94      1.00      0.97      4505
        8ASK       0.94      1.00      0.97      4505
        BPSK       1.00      1.00      1.00      4506
        8PSK       1.00      1.00      1.00      4505
   AM-SSB-SC       0.99      1.00      1.00      4506
        4ASK       0.95      1.00      0.97      4505
       16PSK       1.00      1.00      1.00      4506
      64APSK       0.98      1.00      0.99      4506
      128QAM       0.74      1.00      0.85      4505
     128APSK       0.73      0.99      0.84      4506
   AM-DSB-SC       0.57      1.00      0.73      4506
   AM-SSB-WC       0.50      1.00      0.67      4505
       64QAM       0.51      1.00      0.67      4506
        QPSK       0.78      1.00      0.87      4505
      256QAM       0.50      1.00      0.67      4506
   AM-DSB-WC       1.00      1.00      1.00      4506
         OOK       1.00      1.00      1.00      4505
       16QAM       1.00      1.00      1.00      4506

   micro avg       0.83      1.00      0.91    108135
   macro avg       0.88      1.00      0.92    108135
weighted avg       0.88      1.00      0.92    108135
 samples avg       0.90      1.00      0.93    108135
```

```
[ ]:
```