MODULATION TECHNIQUES CLASSIFICATION USING CNN

Submitted by:
Monika(15MI447)
Sapna(15MI448

OBJECTIVE:

To use a convolutional neural network (CNN) for modulation classification.

Modulation is a process of mixing a signal with a sinusoid to produce a new signal. This new signal, conceivably, will have certain benefits over an un-modulated signal. Mixing of low frequency signal with high frequency carrier signal is called modulation.

There are 3 basic types of modulation: Amplitude modulation, Frequency modulation, and Phase modulation:

Amplitude modulation
a type of modulation where the amplitude of the carrier signal is modulated (changed) in proportion to the message signal while the frequency and phase are kept constant.
Frequency modulation
a type of modulation where the frequency of the carrier signal is modulated (changed) in proportion to the message signal while the amplitude and phase are kept constant.
Phase modulation
a type of modulation where the phase of the carrier signal is varied according to the low frequency of the message signal is known as phase modulation.

METHODOLOGY:

1. Generation of variable SNR dataset consisting of different modulations. The data is modified with labeled SNR increments.
   1.1 loading and importing data

```
from google.colab import drive
drive.mount('/content/drive')
filename = "/content/drive/My Drive/RML2016.10a.dict.pkl"
open_file = open(filename,'rb')
data = pickle.load(open_file, encoding='latin1')
```

1.2 loading to numpy arrays
1.3 splitting data
1.4 Creation of tensorflow dataset

2. Every sample is presented using two vectors each of them has 128 elements. The dataset has the signals with raw features so we extracted more features from it which are:

3. First Derivative in Time

4. Integral in Time

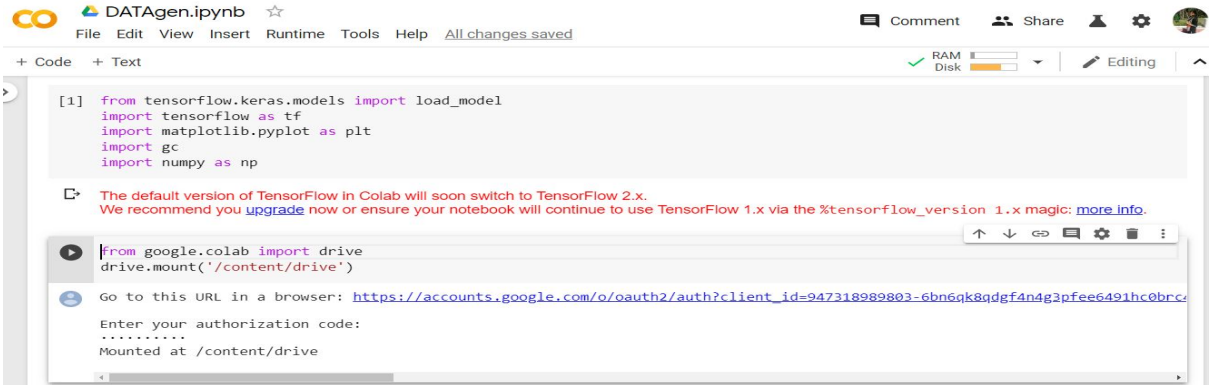
for mod, snr(=18,20) in output

output[(mod,snr)] = output[(mod,snr)][start_indx:fin_indx]

cPickle.dump(X, file("RML2016.10a_dict.pkl", "wb"))
X = np.vstack(X.values())
cPickle.dump(X, file("RML2016.10a_dict.dat", "wb"))

5. We focus on the following end result:
a) Plot of the accuracy against the SNR.
b) Display of the Confusion matrix.

DATA GENERATION:



```
[ ]  gc.collect()
```

```
70
```

```
[ ]  def extract_fn_2(tfrecord):
            # Extract features using the keys set during creation
            features = {
                'feature1': tf.FixedLenFeature([128], tf.float32,),
                'feature2': tf.FixedLenFeature([128], tf.float32),
                'label': tf.FixedLenFeature([10],tf.int64)
            }

            # Extract the data record
            sample = tf.parse_single_example(tfrecord, features)
            data = tf.stack([sample['feature1'], sample['feature2']])
            label = sample['label']
            return [data,label]
```

```
[ ]  def extract_fn_4(tfrecord):
            # Extract features using the keys set during creation
            features = {
                'feature1': tf.FixedLenFeature([128], tf.float32),
                'feature2': tf.FixedLenFeature([128], tf.float32),
                'feature3': tf.FixedLenFeature([128], tf.float32),
                'feature4': tf.FixedLenFeature([128], tf.float32),
                'label': tf.FixedLenFeature([10],tf.int64)
            }

            # Extract the data record
            sample = tf.parse_single_example(tfrecord, features)
            data = tf.stack([sample['feature1'], sample['feature2'], sample['feature3'], sample['feature4']])
            label = sample['label']
            return [data,label]
```

```
[ ]  def dataset_tonumpy(dataset,dim):
       #splitting datasets into X and Y
       X = []
       Y = []
       it= dataset.make_one_shot_iterator().get_next()
       sess = tf.Session()
       with tf.train.MonitoredTrainingSession() as sess:
         while not sess.should_stop():
                 data =sess.run(it)
                 X.append(data[0])
                 Y.append(data[1])


       testX = np.array(X).reshape(600000,dim,128)
       testY = np.array(Y).reshape(600000,10)
       return testX,testY
     def plot_confusion_matrix(cm, title='Confusion matrix', cmap=plt.cm.Blues, labels=[]):
         plt.imshow(cm, interpolation='nearest', cmap=cmap)
         plt.title(title)
         plt.colorbar()
```

```
tick_marks = np.arange(len(labels))
plt.xticks(tick_marks, labels, rotation=45)
plt.yticks(tick_marks, labels)
plt.tight_layout()
plt.ylabel('True label')
plt.xlabel('Predicted label')
plt.show()
```

NETWORK LAYERS:

The layers used are:

1. Dense Layer
2. Convolutional 2D Layer
3. Flatten Layer
4. LSTM
5. CNN

Dense Layer is just a regular layer of neurons in a neural network. Each neuron receives input from all the neurons in the previous layer, thus densely connected. The layer has a weight matrix W, a bias vector b, and the activations of previous layers.

Convolutional 2D Layer creates a convolution kernel that is wind with layers input which helps produce a tensor of outputs.

Flatten Layer in between the convolutional layer and the fully connected layer, there is a 'Flatten' layer. Flattening transforms a two-dimensional matrix of features into a vector that can be fed into a fully connected neural network classifier

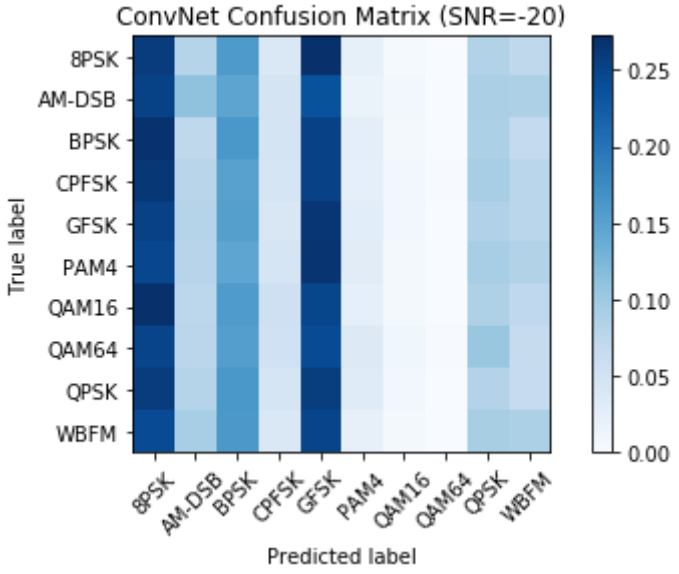
LSTM is an artificial recurrent neural network (RNN) architecture used in the field of deep learning. Unlike standard feedforward neural networks, LSTM has feedback connections. It can not only process single data points (such as images), but also entire sequences of data.

CNN:

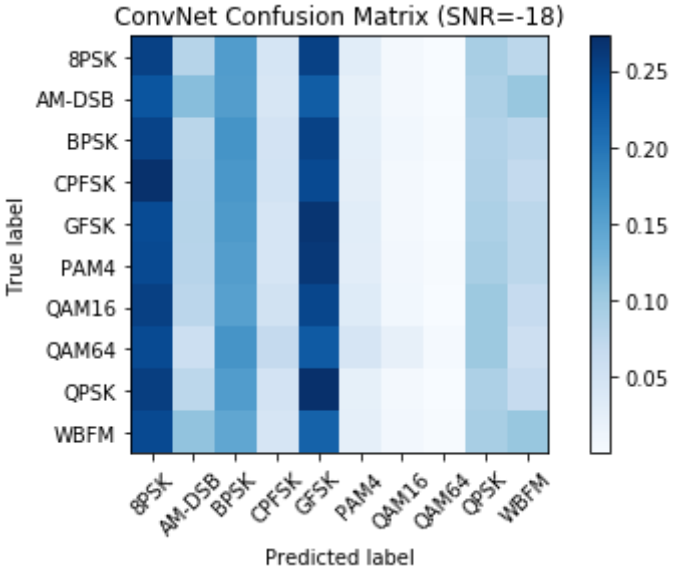
1. The core data structure of Keras is a model. The simplest type of model is the Sequential model, a linear stack of layers.
2. A 3-layer deep neural network consisting only of fully connected layers.

OUTPUT:

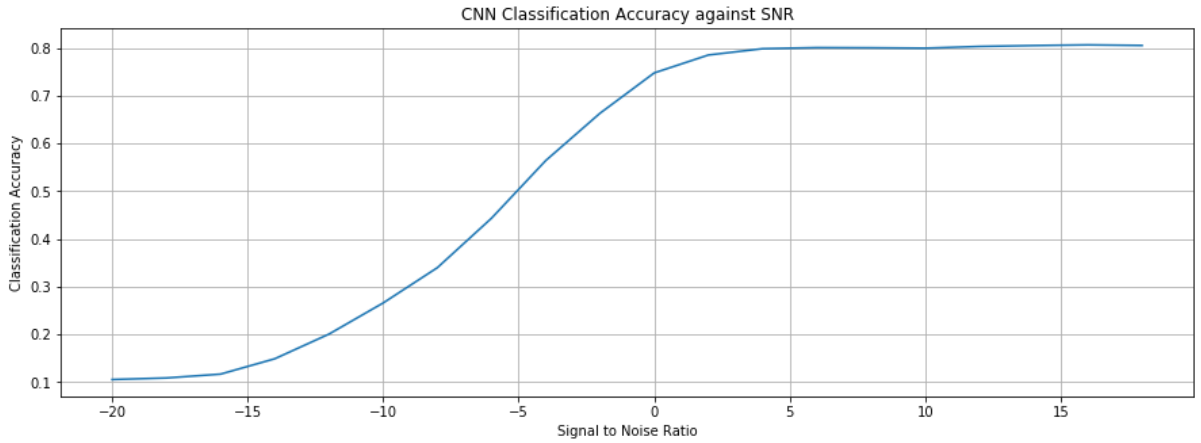
For SNR= 20



For SNR= 18

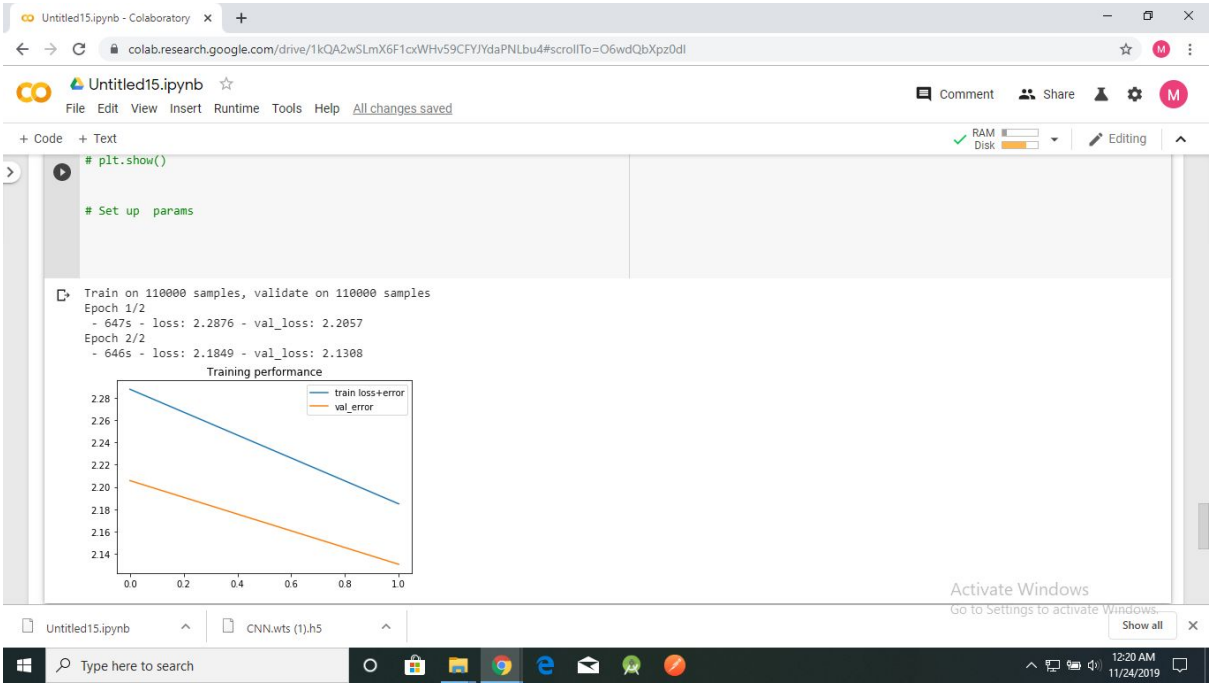


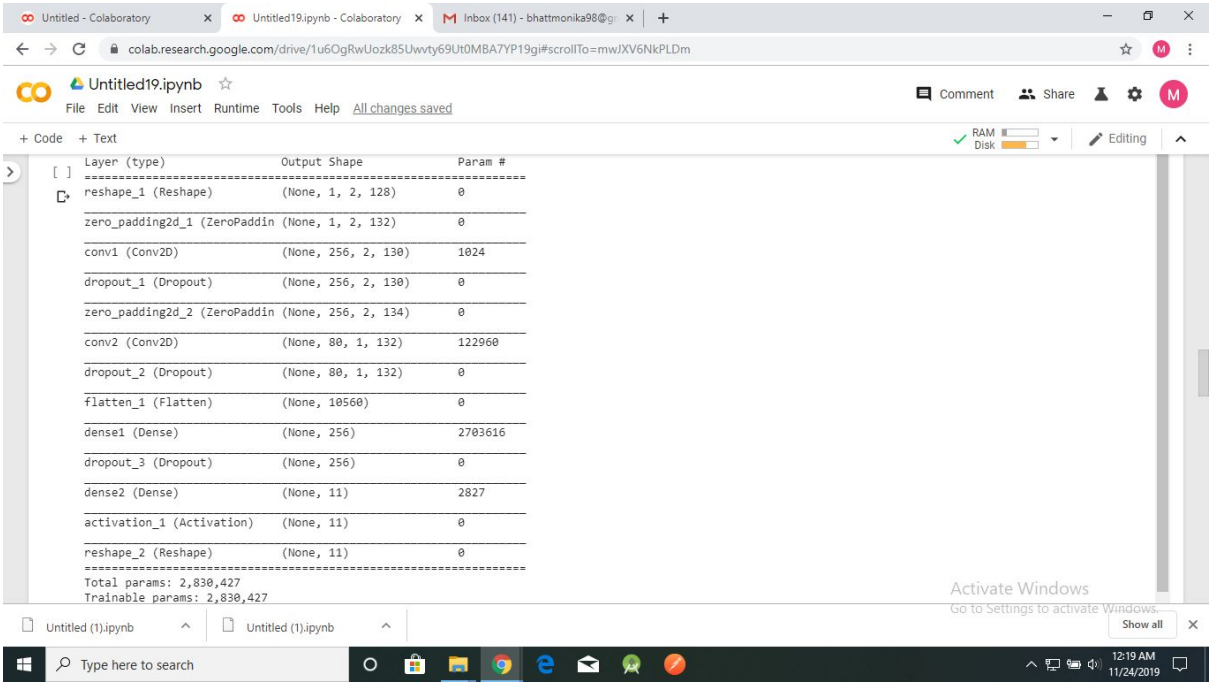
Accuracy Curve:

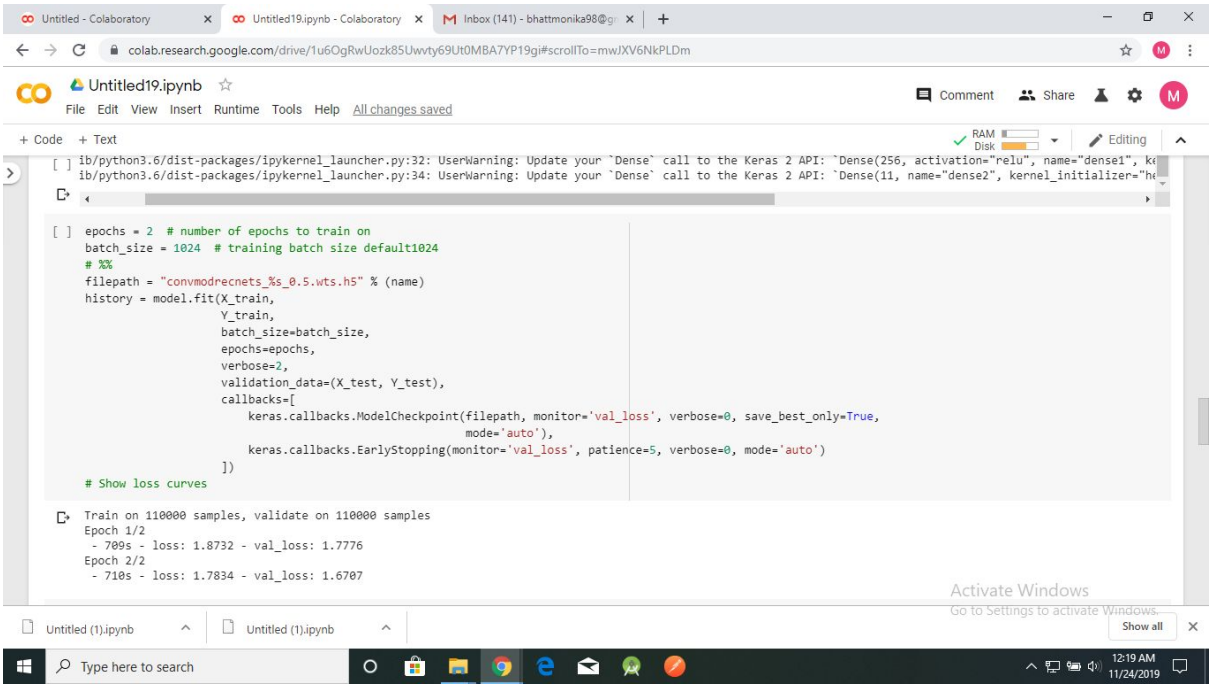


Training Performance Plot & Model Summary:

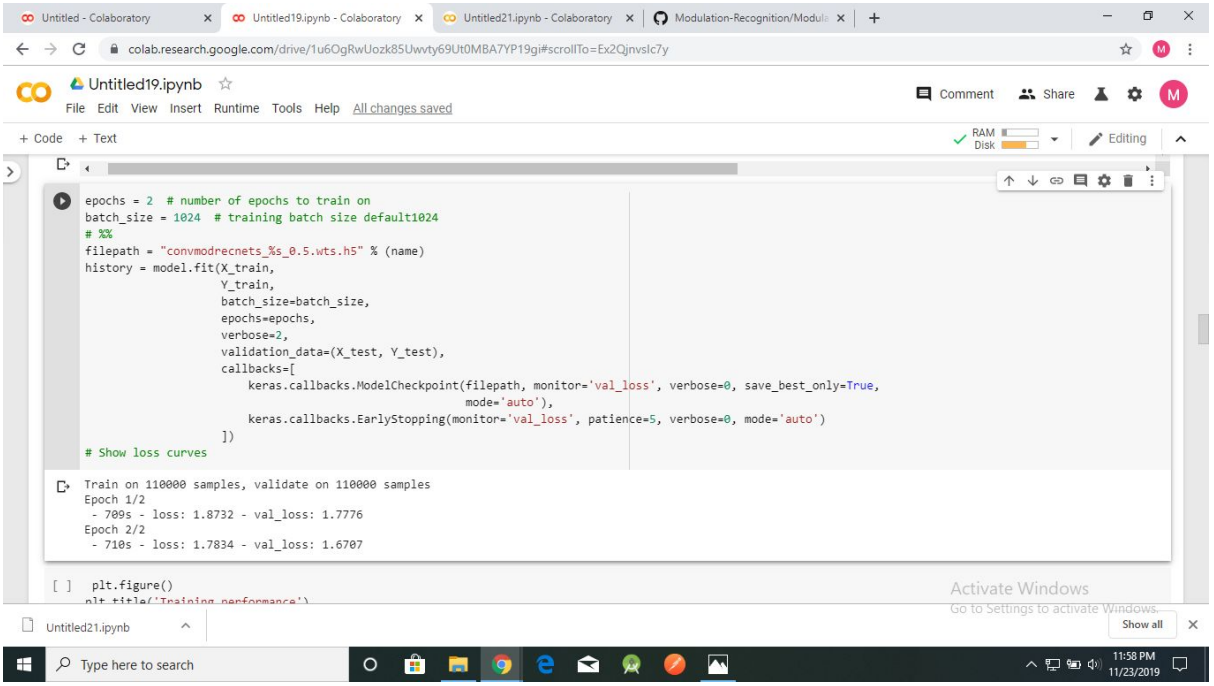
1. Using CNN

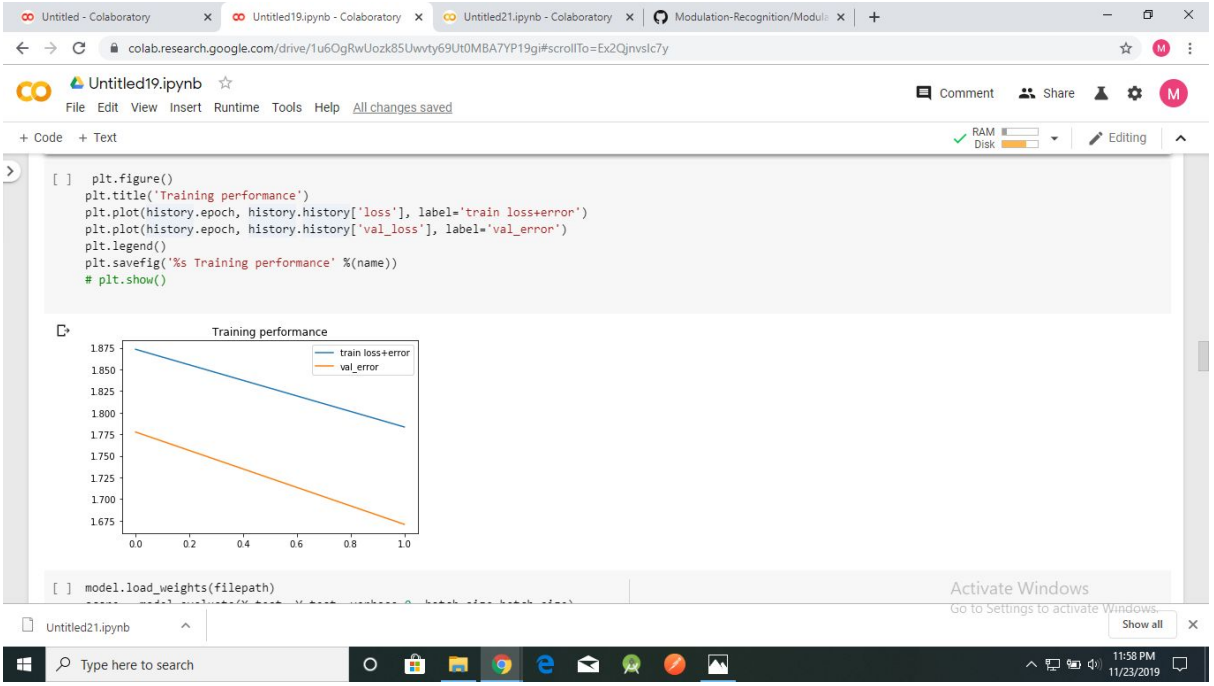


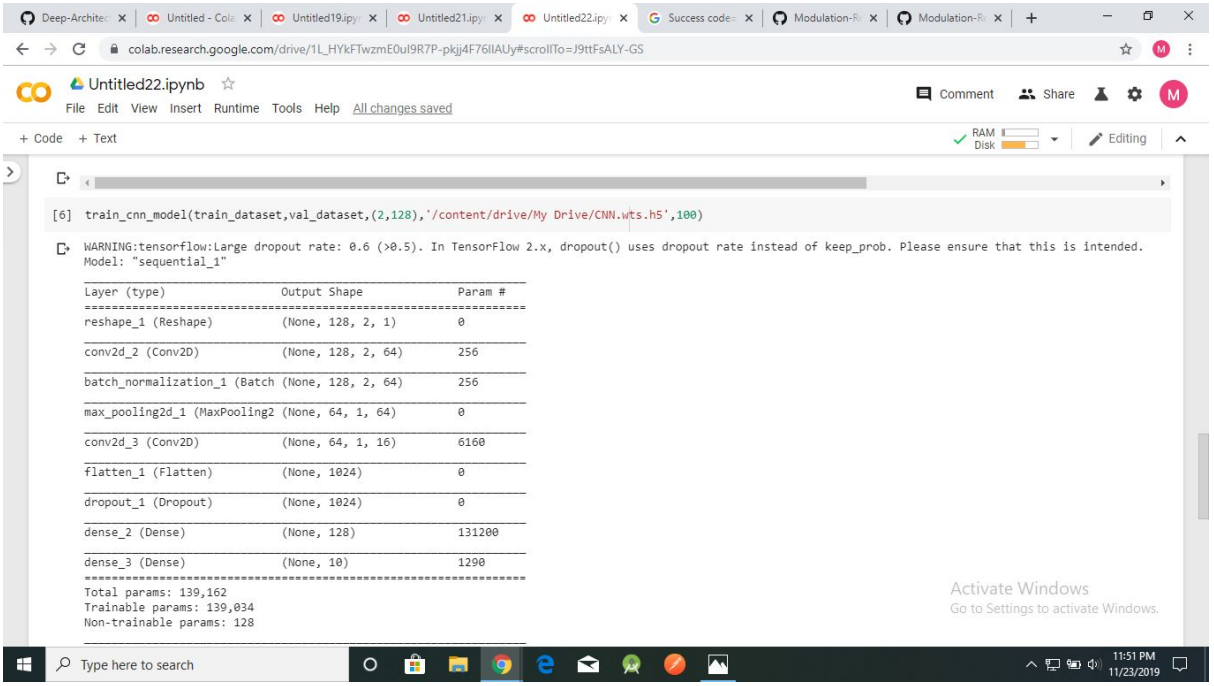

Untitled - Colaboratory
Untitled19.ipynb - Colaboratory
Inbox (141) - bhattmonika98@g
colab.research.google.com/drive/1u6OgRwUozk85Uwvty69Ut0MBA7YP19gi#scrollTo=mwJXV6NkPLDm
Untitled19.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
Editing
Layer (type) Output Shape Param #
reshape_1 (Reshape) (None, 1, 2, 128) 0
zero_padding2d_1 (ZeroPaddin (None, 1, 2, 132) 0
conv1 (Conv2D) (None, 256, 2, 130) 1024
dropout_1 (Dropout) (None, 256, 2, 130) 0
zero_padding2d_2 (ZeroPaddin (None, 256, 2, 134) 0
conv2 (Conv2D) (None, 80, 1, 132) 122960
dropout_2 (Dropout) (None, 80, 1, 132) 0
flatten_1 (Flatten) (None, 10560) 0
dense1 (Dense) (None, 256) 2703616
dropout_3 (Dropout) (None, 256) 0
dense2 (Dense) (None, 11) 2827
activation_1 (Activation) (None, 11) 0
reshape_2 (Reshape) (None, 11) 0
Total params: 2,830,427
Trainable params: 2,830,427
Activate Windows
Go to Settings to activate Windows.
Untitled (1).ipynb
Untitled (1).ipynb
Show all
Type here to search
12:19 AM
11/24/2019


Untitled - Colaboratory
Untitled19.ipynb - Colaboratory
Inbox (141) - bhattmonika98@g
colab.research.google.com/drive/1u6OgRwUozk85Uwvty69Ut0MBA7YP19gi#scrollTo=mwJXV6NkPLDm
Untitled19.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
Editing
ib/python3.6/dist-packages/ipykernel_launcher.py:32: UserWarning: Update your `Dense` call to the Keras 2 API: `Dense(256, activation="relu", name="dense1", ke
ib/python3.6/dist-packages/ipykernel_launcher.py:34: UserWarning: Update your `Dense` call to the Keras 2 API: `Dense(11, name="dense2", kernel_initializer="he
epochs = 2 # number of epochs to train on
batch_size = 1024 # training batch size default1024
# %%
filepath = "convmodrecnets_%s_0.5.wts.h5" % (name)
history = model.fit(X_train,
Y_train,
batch_size=batch_size,
epochs=epochs,
verbose=2,
validation_data=(X_test, Y_test),
callbacks=[
keras.callbacks.ModelCheckpoint(filepath, monitor='val_loss', verbose=0, save_best_only=True,
mode='auto'),
keras.callbacks.EarlyStopping(monitor='val_loss', patience=5, verbose=0, mode='auto')
])
Show loss curves
Train on 110000 samples, validate on 110000 samples
Epoch 1/2
- 709s - loss: 1.8732 - val_loss: 1.7776
Epoch 2/2
- 710s - loss: 1.7834 - val_loss: 1.6707
Activate Windows
Go to Settings to activate Windows.
Untitled (1).ipynb
Untitled (1).ipynb
Show all
Type here to search
12:19 AM
11/24/2019

2. Using LSTM


Untitled - Colaboratory
Untitled19.ipynb - Colaboratory
Untitled21.ipynb - Colaboratory
Modulation-Recognition/Modul
colab.research.google.com/drive/1u6OgRwUozk85Uwvty69Ut0MBA7YP19gi#scrollTo=Ex2Qjnvslc7y
Untitled19.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
Editing
epochs = 2 # number of epochs to train on
batch_size = 1024 # training batch size default1024
# %%
filepath = "convmodrecnets_%s_0.5.wts.h5" % (name)
history = model.fit(X_train,
Y_train,
batch_size=batch_size,
epochs=epochs,
verbose=2,
validation_data=(X_test, Y_test),
callbacks=[
keras.callbacks.ModelCheckpoint(filepath, monitor='val_loss', verbose=0, save_best_only=True,
mode='auto'),
keras.callbacks.EarlyStopping(monitor='val_loss', patience=5, verbose=0, mode='auto')
])
Show loss curves
Train on 110000 samples, validate on 110000 samples
Epoch 1/2
- 709s - loss: 1.8732 - val_loss: 1.7776
Epoch 2/2
- 710s - loss: 1.7834 - val_loss: 1.6707
[ ] plt.figure()
Activate Windows
Go to Settings to activate Windows.
Untitled21.ipynb
Show all
Type here to search
11:58 PM
11/23/2019


Untitled - Colaboratory
Untitled19.ipynb - Colaboratory
Untitled21.ipynb - Colaboratory
Modulation-Recognition/Modul
colab.research.google.com/drive/1u6OgRwUozk85Uwvty69Ut0MBA7YP19gi#scrollTo=Ex2Qjnvslc7y
Untitled19.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
Editing
[ ] plt.figure()
plt.title('Training performance')
plt.plot(history.epoch, history.history['loss'], label='train loss+error')
plt.plot(history.epoch, history.history['val_loss'], label='val_error')
plt.legend()
plt.savefig('%s Training performance' %(name))
# plt.show()
Training performance
train loss+error
val_error
1.875
1.850
1.825
1.800
1.775
1.750
1.725
1.700
1.675
0.0
0.2
0.4
0.6
0.8
1.0
[ ] model.load_weights(filepath)
Activate Windows
Go to Settings to activate Windows.
Untitled21.ipynb
Show all
Type here to search
11:58 PM
11/23/2019


Deep-Architec
Untitled - Cola
Untitled19.ipyn
Untitled21.ipyn
Untitled22.ipyn
Success code=
Modulation-R
Modulation-R
colab.research.google.com/drive/1L_HYkFTwzmE0uI9R7P-pkjj4F76IIAUy#scrollTo=J9ttFsALY-GS
Untitled22.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
M
+ Code + Text
RAM
Disk
Editing
[6] train_cnn_model(train_dataset,val_dataset,(2,128),'/content/drive/My Drive/CNN.wts.h5',100)
WARNING:tensorflow:Large dropout rate: 0.6 (>0.5). In TensorFlow 2.x, dropout() uses dropout rate instead of keep_prob. Please ensure that this is intended.
Model: "sequential_1"
Layer (type) Output Shape Param #
reshape_1 (Reshape) (None, 128, 2, 1) 0
conv2d_2 (Conv2D) (None, 128, 2, 64) 256
batch_normalization_1 (Batch (None, 128, 2, 64) 256
max_pooling2d_1 (MaxPooling2 (None, 64, 1, 64) 0
conv2d_3 (Conv2D) (None, 64, 1, 16) 6160
flatten_1 (Flatten) (None, 1024) 0
dropout_1 (Dropout) (None, 1024) 0
dense_2 (Dense) (None, 128) 131200
dense_3 (Dense) (None, 10) 1290
Total params: 139,162
Trainable params: 139,034
Non-trainable params: 128
Activate Windows
Go to Settings to activate Windows.
Type here to search
11:51 PM
11/23/2019