Downloading Dataset and structure:

A dictionary having 162060 samples each of 2 vectors each of length 128 representing time series representation of signals generated by GNU radio under different modulation techniques with differents signal to noise ratios (SNRs).

Problem Statement

Given test samples with the same structure classify the samples correctly using deep neural networks and CNNs showing accuracy of prediction of modulation type of each sample and show accuracy of prediction in comparison with amount of noise in the signal .

Features:

Different types of features are used: first, we use raw data of time series of signals; second, features are the first drivitative of the data; third, features are the integrals and the fourth features are a combination of the past 3 ones.

Raw time series data:

We will detect signals from their time representation showing value of each signal at a given time where each sample is represented by two vectors one for Quadrature and other for In-phase representation which is a way of representing signals in digital signal processing.

First Derivative:

Since data is time series representation then we can get the derivative by getting the difference of each two consecutive samples for each channel separately and this generates the derivative. This is correct because the derivative is the slope of the tangent by definition and the difference between each two values is one second.

Integral:

We will apply also integration process to our data-set by using scipy cumulative trapezoidal rule integration method as shown in the code below:

```
data_int=integrate.cumtrapz(data, initial=0)
```

In general, using same parameters (raw data case) found at setting our NN and applying them on the integrated version of data-set results in lower accuracy, about 50% maximum accuracy resulted @ 5 layers of NN, 256 input size for each layer, and batch size of 100.

Combined Data:

The data is combined by making more channels, in the previous features we used only 2 channels but now we will use 6 channels two for each type of the first 3 feature types.
This will allow our NN to have more features to train on and hopefully more accuracy.

Differentiation features as a detailed use case:

One hidden layer (dense):

Having only one hidden layer with number of units {64,128,256} we observed that as we increase number of units accuracy increases (training, validation and testing).
But it is still low compared to multiple layered models with a maximum accuracy achieved of 21%.

Two hidden layers (dense):

Having two hidden layers with different sets of number of units,

1. We tried making all layers with the same number of units and then made them different trying all combinations to make sense which combinations give best accuracy.
2. We also tried different batch sizes to see their effect.
3. We fixed number of epochs to 100

Conclusions:

- two layers gives better accuracy than one hidden layer with a minimum test accuracy of 44% and maximum of 47.4%

- as we increase batch size number of epochs decrease , running time decreases and accuracy increases (also memory usage increases but we are using colab so it doesn't matter to us)
- We settled with 1000 which is 1/16 of number of samples

- increasing number of units per layer from 128 to 256 increased accuracy by about 0.7% with fixing all other parameters and making number of units the same for 2 layers

- by making one layer 128 units and other one 256 units we notice that accuracy decreases by about 3% than having both of 256 units
- This may mean that having both layers with same number of units increases accuracy or means that as we increase number of units per layer in general accuracy increases

Three hidden layers (dense):

Having two hidden layers with different sets of number of units,

1. least accuracy achieved is 40.5% and maximum accuracy is 47.5% which is almost equivalent to the maximum in two hidden layer model. This means that on increasing another layer we didn't add anything but complexity
2. in general, better results occur when we increase number of units in layers and occur significantly when we make last layer bigger than others, regarding units, and 1st bigger than those in the middle

Four hidden layers (dense):

1. Improved average accuracy by about 3% which is not much compared to complexity added
2. As we increase number of units accuracy tends to increase by a few numbers
3. Maximum accuracy is 46.5% which is less than that of three hidden layers
4. As we increase more layers accuracy doesn't get much better but adds complexity

Raw Data features:

Using Raw data time series representation we observe the following:

1. best accuracy is 64.97% having 4 hidden layers
2. best accuracies at high batch size 700-1000

Integrated Data features:

Using integrated data time series representation we observe the following:

1. best accuracy is 51% having 4 hidden layers
2. best accuracies at high batch size 700-1000
3. CNNs with accuracy 72%

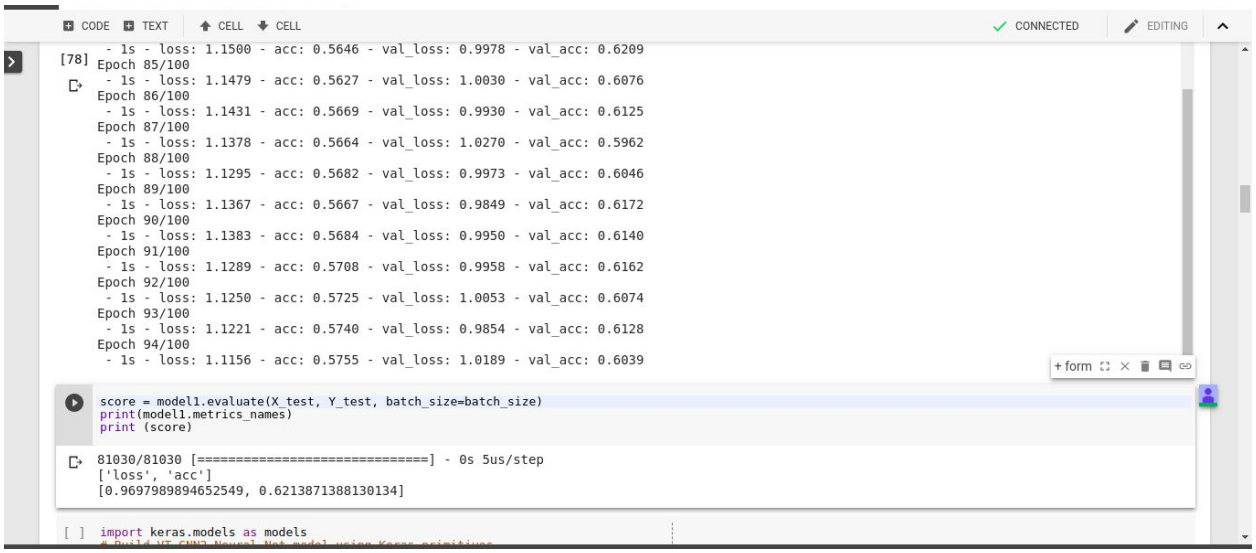
Combined features:

1. best accuracy is 51% using 4 hidden layers
2. high batch size of 700

-- Page left blank intentionally --

Screenshots:

Raw Data tuning:



```
CODE  TEXT  CELL  CELL                                                    CONNECTED  EDITING

[78]  - 1s - loss: 1.1500 - acc: 0.5646 - val_loss: 0.9978 - val_acc: 0.6209
      Epoch 85/100
       - 1s - loss: 1.1479 - acc: 0.5627 - val_loss: 1.0030 - val_acc: 0.6076
      Epoch 86/100
       - 1s - loss: 1.1431 - acc: 0.5669 - val_loss: 0.9930 - val_acc: 0.6125
      Epoch 87/100
       - 1s - loss: 1.1378 - acc: 0.5664 - val_loss: 1.0270 - val_acc: 0.5962
      Epoch 88/100
       - 1s - loss: 1.1295 - acc: 0.5682 - val_loss: 0.9973 - val_acc: 0.6046
      Epoch 89/100
       - 1s - loss: 1.1367 - acc: 0.5667 - val_loss: 0.9849 - val_acc: 0.6172
      Epoch 90/100
       - 1s - loss: 1.1383 - acc: 0.5684 - val_loss: 0.9950 - val_acc: 0.6140
      Epoch 91/100
       - 1s - loss: 1.1289 - acc: 0.5708 - val_loss: 0.9958 - val_acc: 0.6162
      Epoch 92/100
       - 1s - loss: 1.1250 - acc: 0.5725 - val_loss: 1.0053 - val_acc: 0.6074
      Epoch 93/100
       - 1s - loss: 1.1221 - acc: 0.5740 - val_loss: 0.9854 - val_acc: 0.6128
      Epoch 94/100
       - 1s - loss: 1.1156 - acc: 0.5755 - val_loss: 1.0189 - val_acc: 0.6039

                                                                          + form

      score = model1.evaluate(X_test, Y_test, batch_size=batch_size)
      print(model1.metrics_names)
      print (score)

      81030/81030 [==============================] - 0s 5us/step
      ['loss', 'acc']
      [0.9697989894652549, 0.6213871388130134]

[ ]   import keras.models as models
```

```
[77]  # Set up some params
      nb_epoch = 100   # number of epochs to train on
      batch_size = 700  # training batch size

[78]  # perform training ...
      #   - call the main training loop in keras for our network+dataset
      filepath = 'convmodrecnets_CNN2_0.5.wts.h5'
      history=model1.fit(X_train,
          Y_train,
          batch_size=batch_size,
          epochs=nb_epoch,
          verbose=2,
           validation_split=0.05,
          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')
          ])
      # we re-load the best weights once training is finished
      model1.load_weights(filepath)

      Epoch 66/100
       - 1s - loss: 1.2392 - acc: 0.5227 - val_loss: 1.0868 - val_acc: 0.5726
      Epoch 67/100
       - 1s - loss: 1.2290 - acc: 0.5285 - val_loss: 1.0821 - val_acc: 0.5772
      Epoch 68/100
       - 1s - loss: 1.2270 - acc: 0.5295 - val_loss: 1.0760 - val_acc: 0.5809
      Epoch 69/100
       - 1s - loss: 1.2174 - acc: 0.5334 - val_loss: 1.0941 - val_acc: 0.5755
      Epoch 70/100
       - 1s - loss: 1.2146 - acc: 0.5358 - val_loss: 1.0562 - val_acc: 0.5814
      Epoch 71/100
       - 1s - loss: 1.2091 - acc: 0.5389 - val_loss: 1.0534 - val_acc: 0.5879
      Epoch 72/100
```

```
Layer (type)                 Output Shape              Param #
=================================================================
reshape_35 (Reshape)         (None, 2, 128, 1)         0
_________________________________________________________________
dropout_80 (Dropout)         (None, 2, 128, 1)         0
_________________________________________________________________
flatten_18 (Flatten)         (None, 256)               0
_________________________________________________________________
dense1 (Dense)               (None, 256)               65792
_________________________________________________________________
dropout_81 (Dropout)         (None, 256)               0
_________________________________________________________________
dense2 (Dense)               (None, 128)               32896
_________________________________________________________________
dropout_82 (Dropout)         (None, 128)               0
_________________________________________________________________
dense3 (Dense)               (None, 128)               16512
_________________________________________________________________
dropout_83 (Dropout)         (None, 128)               0
_________________________________________________________________
dense4 (Dense)               (None, 64)                8256
_________________________________________________________________
dropout_84 (Dropout)         (None, 64)                0
_________________________________________________________________
dense5 (Dense)               (None, 11)                715
_________________________________________________________________
activation_18 (Activation)   (None, 11)                0
_________________________________________________________________
reshape_36 (Reshape)         (None, 11)                0
=================================================================
Total params: 124,171
Trainable params: 124,171
Non-trainable params: 0
```

CODE TEXT CELL CELL CONNECTED EDITING

```
[76] #fully connected neural network
dr = 0.3
model1 = keras.models.Sequential()
model1.add (Reshape(in_shp+[1], input_shape=in_shp))
model1.add (Dropout(dr))
model1.add(Flatten())
model1.add(Dense(256, activation='relu', kernel_initializer='he_normal', name="dense1"))
model1.add(Dropout(dr))
model1.add(Dense(128, activation='relu', kernel_initializer='he_normal', name="dense2"))
model1.add(Dropout(dr))
model1.add(Dense(128, activation='relu', kernel_initializer='he_normal', name="dense3"))
model1.add(Dropout(dr))
model1.add(Dense(64, activation='relu', kernel_initializer='he_normal', name="dense4"))
model1.add(Dropout(dr))
model1.add(Dense( len(classes), kernel_initializer='he_normal', name="dense5" ))
model1.add(Activation('softmax'))
model1.add(Reshape([len(classes)]))
model1.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
model1.summary()
```

```
Layer (type)                 Output Shape              Param #
=================================================================
reshape_35 (Reshape)         (None, 2, 128, 1)         0
_________________________________________________________________
dropout_80 (Dropout)         (None, 2, 128, 1)         0
_________________________________________________________________
flatten_18 (Flatten)         (None, 256)               0
_________________________________________________________________
dense1 (Dense)               (None, 256)               65792
_________________________________________________________________
dropout_81 (Dropout)         (None, 256)               0
_________________________________________________________________
dense2 (Dense)               (None, 128)               32896
```

CODE TEXT CELL CELL CONNECTED EDITING

```
reshape_32 (Reshape)         (None, 11)                0
=================================================================
Total params: 133,131
Trainable params: 133,131
Non-trainable params: 0
_________________________________________________________________
```

```
[69] # Set up some params
nb_epoch = 100   # number of epochs to train on
batch_size = 700  # training batch size
```

```
[70] # perform training ...
#   - call the main training loop in keras for our network+dataset
filepath = 'convmodrecnets_CNN2_0.5.wts.h5'
history=model1.fit(X_train,
    Y_train,
    batch_size=batch_size,
    epochs=nb_epoch,
    verbose=2,
     validation_split=0.05,
    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')
    ])
# we re-load the best weights once training is finished
model1.load_weights(filepath)
```

```
Epoch 23/100
 - 1s - loss: 1.1366 - acc: 0.5608 - val_loss: 1.0823 - val_acc: 0.5745
Epoch 24/100
 - 1s - loss: 1.1183 - acc: 0.5684 - val_loss: 1.0723 - val_acc: 0.5871
Epoch 25/100
 - 1s - loss: 1.1006 - acc: 0.5769 - val_loss: 1.0543 - val_acc: 0.5896
Epoch 26/100
```

CODE TEXT CELL CELL CONNECTED EDITING

[68]
```
Layer (type)                 Output Shape              Param #
=================================================================
reshape_31 (Reshape)         (None, 2, 128, 1)         0
_________________________________________________________________
dropout_70 (Dropout)         (None, 2, 128, 1)         0
_________________________________________________________________
flatten_16 (Flatten)         (None, 256)               0
_________________________________________________________________
dense1 (Dense)               (None, 256)               65792
_________________________________________________________________
dropout_71 (Dropout)         (None, 256)               0
_________________________________________________________________
dense2 (Dense)               (None, 128)               32896
_________________________________________________________________
dropout_72 (Dropout)         (None, 128)               0
_________________________________________________________________
dense3 (Dense)               (None, 128)               16512
_________________________________________________________________
dropout_73 (Dropout)         (None, 128)               0
_________________________________________________________________
dense4 (Dense)               (None, 128)               16512
_________________________________________________________________
dropout_74 (Dropout)         (None, 128)               0
_________________________________________________________________
dense5 (Dense)               (None, 11)                1419
_________________________________________________________________
activation_16 (Activation)   (None, 11)                0
_________________________________________________________________
reshape_32 (Reshape)         (None, 11)                0
=================================================================
Total params: 133,131
Trainable params: 133,131
Non-trainable params: 0
```

CODE TEXT CELL CELL CONNECTED EDITING

```
Trainable params: 173,451
Non-trainable params: 0
_________________________________________________________________
```

[58]
```
# Set up some params
nb_epoch = 100   # number of epochs to train on
batch_size = 1000  # training batch size
```

[59]
```
# perform training ...
#   - call the main training loop in keras for our network+dataset
filepath = 'convmodrecnets_CNN2_0.5.wts.h5'
history=model1.fit(X_train,
    Y_train,
    batch_size=batch_size,
    epochs=nb_epoch,
    verbose=2,
     validation_split=0.05,
    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')
    ])
# we re-load the best weights once training is finished
model1.load_weights(filepath)
```

```
Train on 76978 samples, validate on 4052 samples
Epoch 1/50
 - 2s - loss: 3.0463 - acc: 0.1765 - val_loss: 2.2358 - val_acc: 0.2078
Epoch 2/50
 - 1s - loss: 2.2402 - acc: 0.2054 - val_loss: 2.1879 - val_acc: 0.2127
Epoch 3/50
 - 1s - loss: 2.1942 - acc: 0.2109 - val_loss: 2.1469 - val_acc: 0.2122
Epoch 4/50
 - 1s - loss: 2.1511 - acc: 0.2120 - val_loss: 2.1033 - val_acc: 0.2394
Epoch 5/50
```

CODE TEXT CELL CELL CONNECTED EDITING

```
#fully connected neural network
dr = 0.1
model1 = keras.models.Sequential()
model1.add (Reshape(in_shp+[1], input_shape=in_shp))
model1.add (Dropout(dr))
model1.add(Flatten())
model1.add(Dense(256, activation='relu', kernel_initializer='he_normal', name="dense1"))
model1.add(Dropout(dr))
model1.add(Dense(256, activation='relu', kernel_initializer='he_normal', name="dense2"))
model1.add(Dropout(dr))
model1.add(Dense(128, activation='relu', kernel_initializer='he_normal', name="dense3"))
model1.add(Dropout(dr))
model1.add(Dense(64, activation='relu', kernel_initializer='he_normal', name="dense4"))
model1.add(Dropout(dr))
model1.add(Dense( len(classes), kernel_initializer='he_normal', name="dense5" ))
model1.add(Activation('softmax'))
model1.add(Reshape([len(classes)]))
model1.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
model1.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
reshape_29 (Reshape)         (None, 2, 128, 1)         0
_________________________________________________________________
dropout_65 (Dropout)         (None, 2, 128, 1)         0
_________________________________________________________________
flatten_15 (Flatten)         (None, 256)               0
_________________________________________________________________
dense1 (Dense)               (None, 256)               65792
_________________________________________________________________
dropout_66 (Dropout)         (None, 256)               0
_________________________________________________________________
dense2 (Dense)               (None, 256)               65792
```

```
 - 1s - loss: 1.0688 - acc: 0.5928 - val_loss: 1.0394 - val_acc: 0.5962
[59] Epoch 44/50
 - 1s - loss: 1.0545 - acc: 0.5984 - val_loss: 1.0290 - val_acc: 0.5982
Epoch 45/50
 - 1s - loss: 1.0363 - acc: 0.6032 - val_loss: 1.0190 - val_acc: 0.6160
Epoch 46/50
 - 1s - loss: 1.0276 - acc: 0.6083 - val_loss: 1.0133 - val_acc: 0.6115
Epoch 47/50
 - 1s - loss: 1.0217 - acc: 0.6084 - val_loss: 0.9989 - val_acc: 0.6150
Epoch 48/50
 - 1s - loss: 1.0085 - acc: 0.6123 - val_loss: 0.9953 - val_acc: 0.6162
Epoch 49/50
 - 1s - loss: 1.0018 - acc: 0.6167 - val_loss: 0.9910 - val_acc: 0.6153
Epoch 50/50
 - 1s - loss: 0.9946 - acc: 0.6174 - val_loss: 0.9885 - val_acc: 0.6212

[60] score = model1.evaluate(X_test, Y_test, batch_size=batch_size)
     print(model1.metrics_names)
     print (score)

     81030/81030 [==============================] - 0s 4us/step
     ['loss', 'acc']
     [0.9838209509524051, 0.6214365039340757]

[ ]  import keras.models as models
     # Build VT-CNN2 Neural Net model using Keras primitives --
     #  - Reshape [N,2,128] to [N,1,2,128] on input
     #  - Pass through 2 2DConv/ReLu layers
     #  - Pass through 2 Dense layers (ReLu and Softmax)
     #  - Perform categorical cross entropy optimization

     dr = 0.1 # dropout rate (%)
     model = keras.models.Sequential()
```

```
     Epoch 30/50
[55]  - 7s - loss: 0.9508 - acc: 0.6325 - val_loss: 0.9703 - val_acc: 0.6227
     Epoch 31/50
      - 7s - loss: 0.9438 - acc: 0.6321 - val_loss: 0.9743 - val_acc: 0.6199
     Epoch 32/50
      - 7s - loss: 0.9420 - acc: 0.6344 - val_loss: 0.9584 - val_acc: 0.6246
     Epoch 33/50
      - 7s - loss: 0.9384 - acc: 0.6369 - val_loss: 0.9558 - val_acc: 0.6266
     Epoch 34/50
      - 7s - loss: 0.9377 - acc: 0.6374 - val_loss: 0.9558 - val_acc: 0.6251
     Epoch 35/50
      - 7s - loss: 0.9344 - acc: 0.6385 - val_loss: 0.9722 - val_acc: 0.6251
     Epoch 36/50
      - 7s - loss: 0.9299 - acc: 0.6395 - val_loss: 0.9640 - val_acc: 0.6251
     Epoch 37/50
      - 7s - loss: 0.9272 - acc: 0.6407 - val_loss: 0.9634 - val_acc: 0.6207
     Epoch 38/50
      - 7s - loss: 0.9253 - acc: 0.6424 - val_loss: 0.9686 - val_acc: 0.6227

[56] score = model1.evaluate(X_test, Y_test, batch_size=batch_size)
     print(model1.metrics_names)
     print (score)

     81030/81030 [==============================] - 3s 32us/step
     ['loss', 'acc']
     [0.9492381242448846, 0.6300999652605821]

[ ]  import keras.models as models
     # Build VT-CNN2 Neural Net model using Keras primitives --
     #  - Reshape [N,2,128] to [N,1,2,128] on input
     #  - Pass through 2 2DConv/ReLu layers
     #  - Pass through 2 Dense layers (ReLu and Softmax)
```

```
[53] Layer (type)                 Output Shape              Param #
     =================================================================
     reshape_25 (Reshape)         (None, 2, 128, 1)         0
     _________________________________________________________________
     dropout_55 (Dropout)         (None, 2, 128, 1)         0
     _________________________________________________________________
     flatten_13 (Flatten)         (None, 256)               0
     _________________________________________________________________
     dense1 (Dense)               (None, 256)               65792
     _________________________________________________________________
     dropout_56 (Dropout)         (None, 256)               0
     _________________________________________________________________
     dense2 (Dense)               (None, 128)               32896
     _________________________________________________________________
     dropout_57 (Dropout)         (None, 128)               0
     _________________________________________________________________
     dense3 (Dense)               (None, 64)                8256
     _________________________________________________________________
     dropout_58 (Dropout)         (None, 64)                0
     _________________________________________________________________
     dense4 (Dense)               (None, 32)                2080
     _________________________________________________________________
     dropout_59 (Dropout)         (None, 32)                0
     _________________________________________________________________
     dense5 (Dense)               (None, 11)                363
     _________________________________________________________________
     activation_13 (Activation)   (None, 11)                0
     _________________________________________________________________
     reshape_26 (Reshape)         (None, 11)                0
     =================================================================
     Total params: 109,387
     Trainable params: 109,387
     Non-trainable params: 0
```

CODE TEXT CELL CELL CONNECTED EDITING

```
[55]  - 7s - loss: 0.9766 - acc: 0.6223 - val_loss: 0.9722 - val_acc: 0.6207
      Epoch 26/50
       - 7s - loss: 0.9690 - acc: 0.6221 - val_loss: 0.9670 - val_acc: 0.6281
      Epoch 27/50
       - 7s - loss: 0.9624 - acc: 0.6277 - val_loss: 0.9664 - val_acc: 0.6182
      Epoch 28/50
       - 7s - loss: 0.9616 - acc: 0.6280 - val_loss: 0.9710 - val_acc: 0.6190
      Epoch 29/50
       - 7s - loss: 0.9505 - acc: 0.6311 - val_loss: 0.9698 - val_acc: 0.6251
      Epoch 30/50
       - 7s - loss: 0.9508 - acc: 0.6325 - val_loss: 0.9703 - val_acc: 0.6227
      Epoch 31/50
       - 7s - loss: 0.9438 - acc: 0.6321 - val_loss: 0.9743 - val_acc: 0.6199
      Epoch 32/50
       - 7s - loss: 0.9420 - acc: 0.6344 - val_loss: 0.9584 - val_acc: 0.6246
      Epoch 33/50
       - 7s - loss: 0.9384 - acc: 0.6369 - val_loss: 0.9558 - val_acc: 0.6266
      Epoch 34/50
       - 7s - loss: 0.9377 - acc: 0.6374 - val_loss: 0.9558 - val_acc: 0.6251
      Epoch 35/50
       - 7s - loss: 0.9344 - acc: 0.6385 - val_loss: 0.9722 - val_acc: 0.6251
      Epoch 36/50
       - 7s - loss: 0.9299 - acc: 0.6395 - val_loss: 0.9640 - val_acc: 0.6251
      Epoch 37/50
       - 7s - loss: 0.9272 - acc: 0.6407 - val_loss: 0.9634 - val_acc: 0.6207
      Epoch 38/50
       - 7s - loss: 0.9253 - acc: 0.6424 - val_loss: 0.9686 - val_acc: 0.6227
```

```
[56]  score = model1.evaluate(X_test, Y_test, batch_size=batch_size)
      print(model1.metrics_names)
      print (score)
```

CODE TEXT CELL CELL CONNECTED EDITING

```
[44]  - 6s - loss: 1.0119 - acc: 0.6112 - val_loss: 1.0169 - val_acc: 0.6091
      Epoch 25/50
       - 6s - loss: 1.0038 - acc: 0.6155 - val_loss: 1.0072 - val_acc: 0.6096
      Epoch 26/50
       - 6s - loss: 0.9979 - acc: 0.6167 - val_loss: 0.9939 - val_acc: 0.6172
      Epoch 27/50
       - 6s - loss: 0.9923 - acc: 0.6177 - val_loss: 1.0049 - val_acc: 0.6066
      Epoch 28/50
       - 6s - loss: 0.9921 - acc: 0.6189 - val_loss: 0.9905 - val_acc: 0.6101
      Epoch 29/50
       - 6s - loss: 0.9836 - acc: 0.6232 - val_loss: 0.9865 - val_acc: 0.6153
      Epoch 30/50
       - 6s - loss: 0.9783 - acc: 0.6225 - val_loss: 0.9904 - val_acc: 0.6172
      Epoch 31/50
       - 6s - loss: 0.9777 - acc: 0.6262 - val_loss: 0.9920 - val_acc: 0.6130
      Epoch 32/50
       - 6s - loss: 0.9708 - acc: 0.6269 - val_loss: 0.9921 - val_acc: 0.6192
      Epoch 33/50
       - 6s - loss: 0.9703 - acc: 0.6275 - val_loss: 0.9897 - val_acc: 0.6162
      Epoch 34/50
       - 6s - loss: 0.9667 - acc: 0.6296 - val_loss: 0.9938 - val_acc: 0.6177
```

+ form

```
      score = model1.evaluate(X_test, Y_test, batch_size=batch_size)
      print(model1.metrics_names)
      print (score)
```

```
      81030/81030 [==============================] - 2s 27us/step
      ['loss', 'acc']
      [0.993114006822098, 0.6144267573656601]
```

```
[ ]   import keras.models as models
```

CODE TEXT CELL CELL CONNECTED EDITING

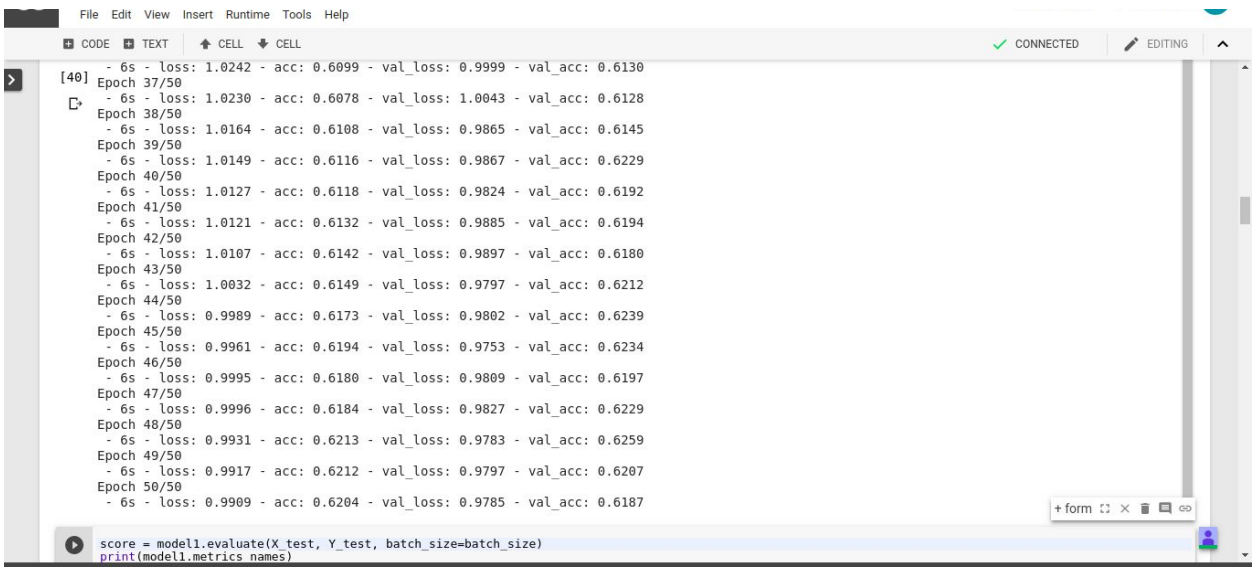
```
[40]  Epoch 43/50
       - 6s - loss: 1.0032 - acc: 0.6149 - val_loss: 0.9797 - val_acc: 0.6212
      Epoch 44/50
       - 6s - loss: 0.9989 - acc: 0.6173 - val_loss: 0.9802 - val_acc: 0.6239
      Epoch 45/50
       - 6s - loss: 0.9961 - acc: 0.6194 - val_loss: 0.9753 - val_acc: 0.6234
      Epoch 46/50
       - 6s - loss: 0.9995 - acc: 0.6180 - val_loss: 0.9809 - val_acc: 0.6197
      Epoch 47/50
       - 6s - loss: 0.9996 - acc: 0.6184 - val_loss: 0.9827 - val_acc: 0.6229
      Epoch 48/50
       - 6s - loss: 0.9931 - acc: 0.6213 - val_loss: 0.9783 - val_acc: 0.6259
      Epoch 49/50
       - 6s - loss: 0.9917 - acc: 0.6212 - val_loss: 0.9797 - val_acc: 0.6207
      Epoch 50/50
       - 6s - loss: 0.9909 - acc: 0.6204 - val_loss: 0.9785 - val_acc: 0.6187
```

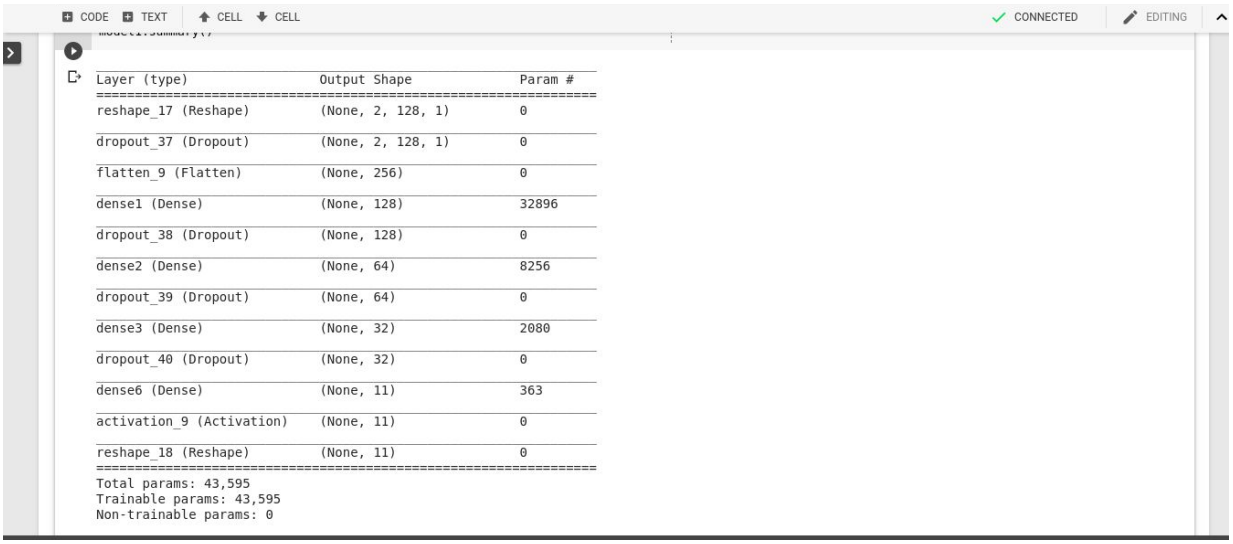
+ form

```
      score = model1.evaluate(X_test, Y_test, batch_size=batch_size)
      print(model1.metrics_names)
      print (score)
```

```
      81030/81030 [==============================] - 2s 31us/step
      ['loss', 'acc']
      [0.9702621066722245, 0.6247686062784646]
```

```
[ ]   import keras.models as models
      # Build VT-CNN2 Neural Net model using Keras primitives --
      #  - Reshape [N,2,128] to [N,1,2,128] on input
      #  - Pass through 2 2DConv/ReLu layers
      #  - Pass through 2 Dense layers (ReLu and Softmax)
      #  - Perform categorical cross entropy optimization
```


File Edit View Insert Runtime Tools Help
CODE TEXT CELL CELL
CONNECTED
EDITING
[40]
- 6s - loss: 1.0242 - acc: 0.6099 - val_loss: 0.9999 - val_acc: 0.6130
Epoch 37/50
- 6s - loss: 1.0230 - acc: 0.6078 - val_loss: 1.0043 - val_acc: 0.6128
Epoch 38/50
- 6s - loss: 1.0164 - acc: 0.6108 - val_loss: 0.9865 - val_acc: 0.6145
Epoch 39/50
- 6s - loss: 1.0149 - acc: 0.6116 - val_loss: 0.9867 - val_acc: 0.6229
Epoch 40/50
- 6s - loss: 1.0127 - acc: 0.6118 - val_loss: 0.9824 - val_acc: 0.6192
Epoch 41/50
- 6s - loss: 1.0121 - acc: 0.6132 - val_loss: 0.9885 - val_acc: 0.6194
Epoch 42/50
- 6s - loss: 1.0107 - acc: 0.6142 - val_loss: 0.9897 - val_acc: 0.6180
Epoch 43/50
- 6s - loss: 1.0032 - acc: 0.6149 - val_loss: 0.9797 - val_acc: 0.6212
Epoch 44/50
- 6s - loss: 0.9989 - acc: 0.6173 - val_loss: 0.9802 - val_acc: 0.6239
Epoch 45/50
- 6s - loss: 0.9961 - acc: 0.6194 - val_loss: 0.9753 - val_acc: 0.6234
Epoch 46/50
- 6s - loss: 0.9995 - acc: 0.6180 - val_loss: 0.9809 - val_acc: 0.6197
Epoch 47/50
- 6s - loss: 0.9996 - acc: 0.6184 - val_loss: 0.9827 - val_acc: 0.6229
Epoch 48/50
- 6s - loss: 0.9931 - acc: 0.6213 - val_loss: 0.9783 - val_acc: 0.6259
Epoch 49/50
- 6s - loss: 0.9917 - acc: 0.6212 - val_loss: 0.9797 - val_acc: 0.6207
Epoch 50/50
- 6s - loss: 0.9909 - acc: 0.6204 - val_loss: 0.9785 - val_acc: 0.6187
+ form
score = model1.evaluate(X_test, Y_test, batch_size=batch_size)
print(model1.metrics_names)


CODE TEXT CELL CELL
CONNECTED
EDITING
Layer (type) Output Shape Param #
reshape_17 (Reshape) (None, 2, 128, 1) 0
dropout_37 (Dropout) (None, 2, 128, 1) 0
flatten_9 (Flatten) (None, 256) 0
dense1 (Dense) (None, 128) 32896
dropout_38 (Dropout) (None, 128) 0
dense2 (Dense) (None, 64) 8256
dropout_39 (Dropout) (None, 64) 0
dense3 (Dense) (None, 32) 2080
dropout_40 (Dropout) (None, 32) 0
dense6 (Dense) (None, 11) 363
activation_9 (Activation) (None, 11) 0
reshape_18 (Reshape) (None, 11) 0
Total params: 43,595
Trainable params: 43,595
Non-trainable params: 0

```
dropout_33 (Dropout)         (None, 2, 128, 1)         0
flatten_8 (Flatten)          (None, 256)               0
dense1 (Dense)               (None, 256)               65792
dropout_34 (Dropout)         (None, 256)               0
dense2 (Dense)               (None, 64)                16448
dropout_35 (Dropout)         (None, 64)                0
dense3 (Dense)               (None, 32)                2080
dropout_36 (Dropout)         (None, 32)                0
dense6 (Dense)               (None, 11)                363
activation_8 (Activation)    (None, 11)                0
reshape_16 (Reshape)         (None, 11)                0
=================================================================
Total params: 84,683
Trainable params: 84,683
Non-trainable params: 0
```

```
[31] # Set up some params
     nb_epoch = 50   # number of epochs to train on
     batch_size = 100  # training batch size
```

```
[32] Epoch 39/50
      - 7s - loss: 0.9876 - acc: 0.6220 - val_loss: 0.9711 - val_acc: 0.6264
     Epoch 40/50
      - 8s - loss: 0.9774 - acc: 0.6254 - val_loss: 0.9851 - val_acc: 0.6165
     Epoch 41/50
      - 8s - loss: 0.9764 - acc: 0.6259 - val_loss: 0.9670 - val_acc: 0.6199
     Epoch 42/50
      - 7s - loss: 0.9731 - acc: 0.6261 - val_loss: 0.9534 - val_acc: 0.6306
     Epoch 43/50
      - 8s - loss: 0.9736 - acc: 0.6258 - val_loss: 0.9611 - val_acc: 0.6234
     Epoch 44/50
      - 7s - loss: 0.9655 - acc: 0.6294 - val_loss: 0.9735 - val_acc: 0.6209
     Epoch 45/50
      - 8s - loss: 0.9606 - acc: 0.6308 - val_loss: 0.9644 - val_acc: 0.6227
     Epoch 46/50
      - 8s - loss: 0.9617 - acc: 0.6314 - val_loss: 0.9684 - val_acc: 0.6347
     Epoch 47/50
      - 8s - loss: 0.9624 - acc: 0.6309 - val_loss: 0.9609 - val_acc: 0.6283
```

```
score = model1.evaluate(X_test, Y_test, batch_size=batch_size)
print(model1.metrics_names)
print (score)
```

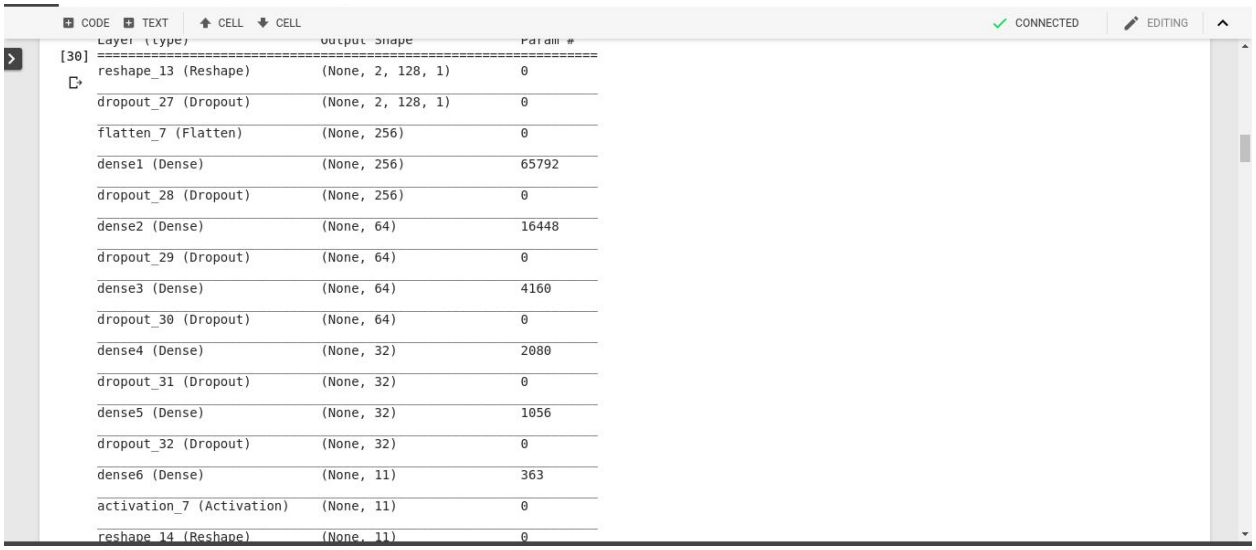
```
81030/81030 [==============================] - 3s 32us/step
['loss', 'acc']
[0.9578799731990985, 0.6293101338640924]
```

```
[ ] import keras.models as models
    # Build VT-CNN2 Neural Net model using Keras primitives --
    #  - Reshape [N,2,128] to [N,1,2,128] on input
    #  - Pass through 2 2DConv/ReLu layers
    #  - Pass through 2 Dense layers (ReLu and Softmax)
```

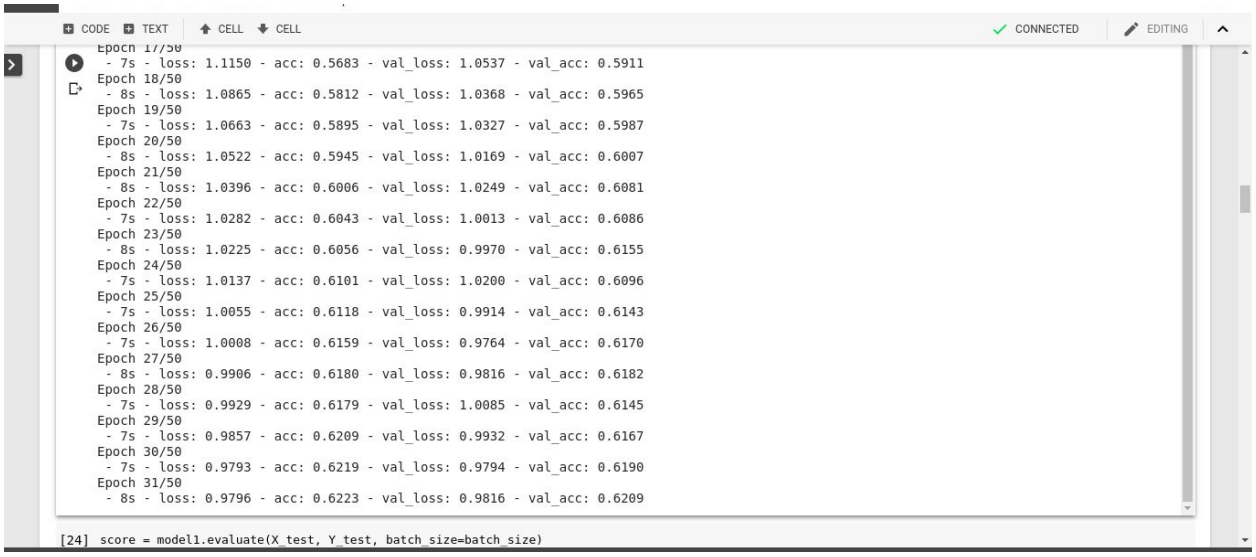
```
[31] # Set up some params
     nb_epoch = 50   # number of epochs to train on
     batch_size = 100  # training batch size
```

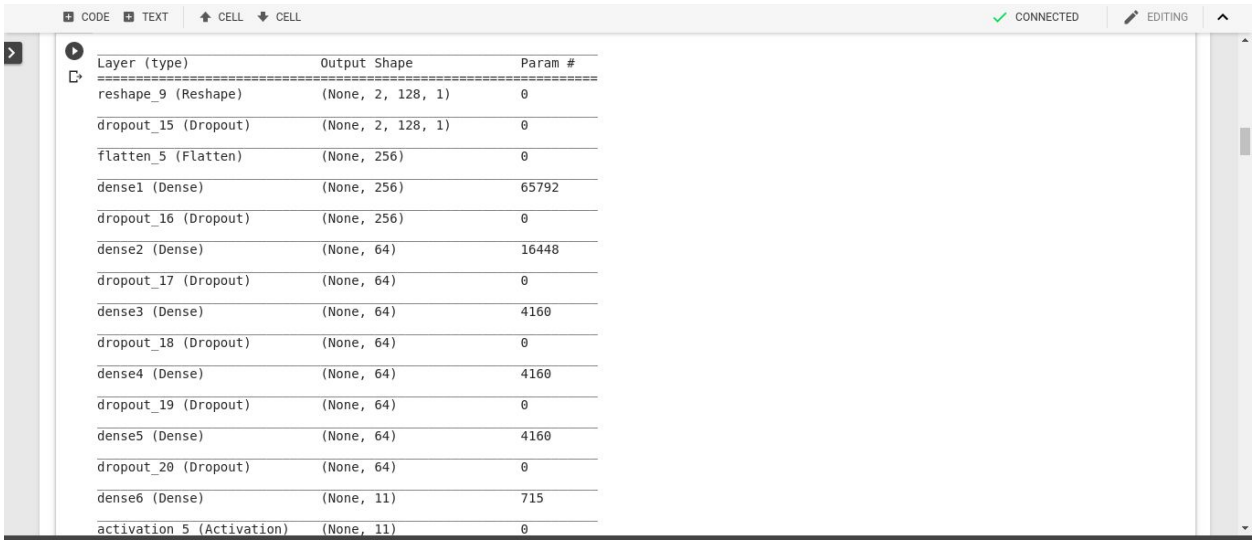
```
# perform training ...
#   - call the main training loop in keras for our network+dataset
filepath = 'convmodrecnets_CNN2_0.5.wts.h5'
history=model1.fit(X_train,
    Y_train,
    batch_size=batch_size,
    epochs=nb_epoch,
    verbose=2,
     validation_split=0.05,
    callbacks = [
        keras.callbacks.ModelCheckpoint(filepath, monitor='val_loss', verbose=0, save_best_only=True, mode='auto'),
        keras.callbacks.EarlyStopping(monitor='val_loss', patience=5, verbose=0, mode='auto')
    ])
# we re-load the best weights once training is finished
model1.load_weights(filepath)
```

```
Train on 76978 samples, validate on 4052 samples
Epoch 1/50
 - 8s - loss: 2.3993 - acc: 0.1863 - val_loss: 2.0778 - val_acc: 0.2110
Epoch 2/50
 - 7s - loss: 1.9942 - acc: 0.2577 - val_loss: 1.8586 - val_acc: 0.2747
Epoch 3/50
 - 8s - loss: 1.8152 - acc: 0.3080 - val_loss: 1.6678 - val_acc: 0.3736
Epoch 4/50
 - 8s - loss: 1.6798 - acc: 0.3739 - val_loss: 1.5821 - val_acc: 0.4158
Epoch 5/50
```

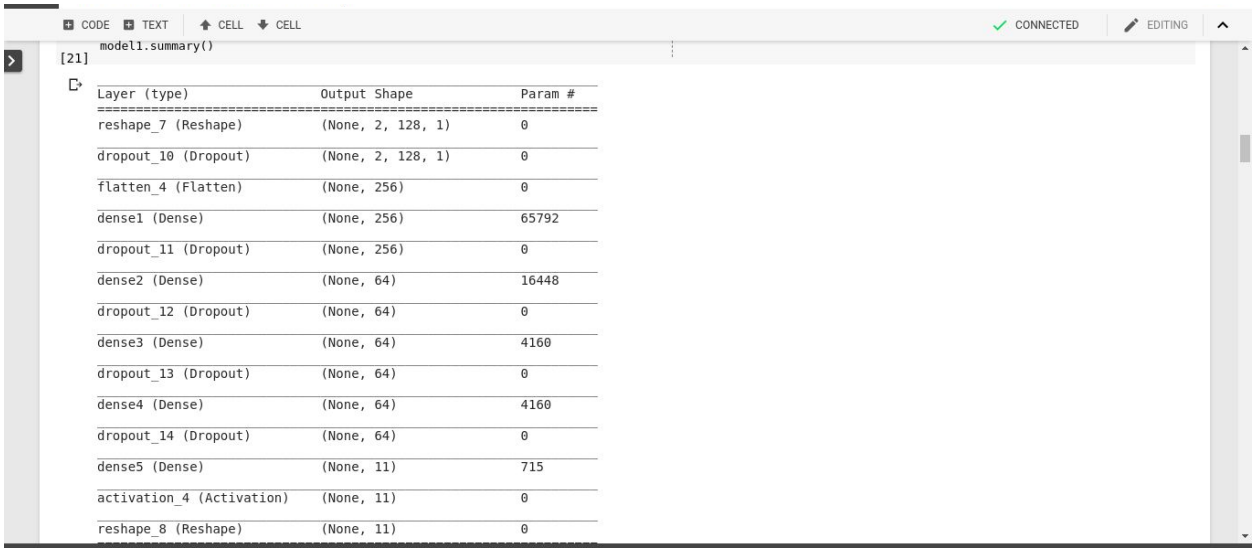

CODE
TEXT
CELL
CELL
CONNECTED
EDITING
[30]
Layer (type) Output Shape Param #
reshape_13 (Reshape) (None, 2, 128, 1) 0
dropout_27 (Dropout) (None, 2, 128, 1) 0
flatten_7 (Flatten) (None, 256) 0
dense1 (Dense) (None, 256) 65792
dropout_28 (Dropout) (None, 256) 0
dense2 (Dense) (None, 64) 16448
dropout_29 (Dropout) (None, 64) 0
dense3 (Dense) (None, 64) 4160
dropout_30 (Dropout) (None, 64) 0
dense4 (Dense) (None, 32) 2080
dropout_31 (Dropout) (None, 32) 0
dense5 (Dense) (None, 32) 1056
dropout_32 (Dropout) (None, 32) 0
dense6 (Dense) (None, 11) 363
activation_7 (Activation) (None, 11) 0
reshape_14 (Reshape) (None, 11) 0


CODE
TEXT
CELL
CELL
CONNECTED
EDITING
[27]
Epoch 22/50
- 7s - loss: 1.0282 - acc: 0.6043 - val_loss: 1.0013 - val_acc: 0.6086
Epoch 23/50
- 8s - loss: 1.0225 - acc: 0.6056 - val_loss: 0.9970 - val_acc: 0.6155
Epoch 24/50
- 7s - loss: 1.0137 - acc: 0.6101 - val_loss: 1.0200 - val_acc: 0.6096
Epoch 25/50
- 7s - loss: 1.0055 - acc: 0.6118 - val_loss: 0.9914 - val_acc: 0.6143
Epoch 26/50
- 7s - loss: 1.0008 - acc: 0.6159 - val_loss: 0.9764 - val_acc: 0.6170
Epoch 27/50
- 8s - loss: 0.9906 - acc: 0.6180 - val_loss: 0.9816 - val_acc: 0.6182
Epoch 28/50
- 7s - loss: 0.9929 - acc: 0.6179 - val_loss: 1.0085 - val_acc: 0.6145
Epoch 29/50
- 7s - loss: 0.9857 - acc: 0.6209 - val_loss: 0.9932 - val_acc: 0.6167
Epoch 30/50
- 7s - loss: 0.9793 - acc: 0.6219 - val_loss: 0.9794 - val_acc: 0.6190
Epoch 31/50
- 8s - loss: 0.9796 - acc: 0.6223 - val_loss: 0.9816 - val_acc: 0.6209
+ form
score = model1.evaluate(X_test, Y_test, batch_size=batch_size)
print(model1.metrics_names)
print (score)
81030/81030 [==============================] - 2s 27us/step
['loss', 'acc']
[0.9702057201332777, 0.623596200760163]
[ ]
import keras.models as models
# Build VT-CNN2 Neural Net model using Keras primitives --

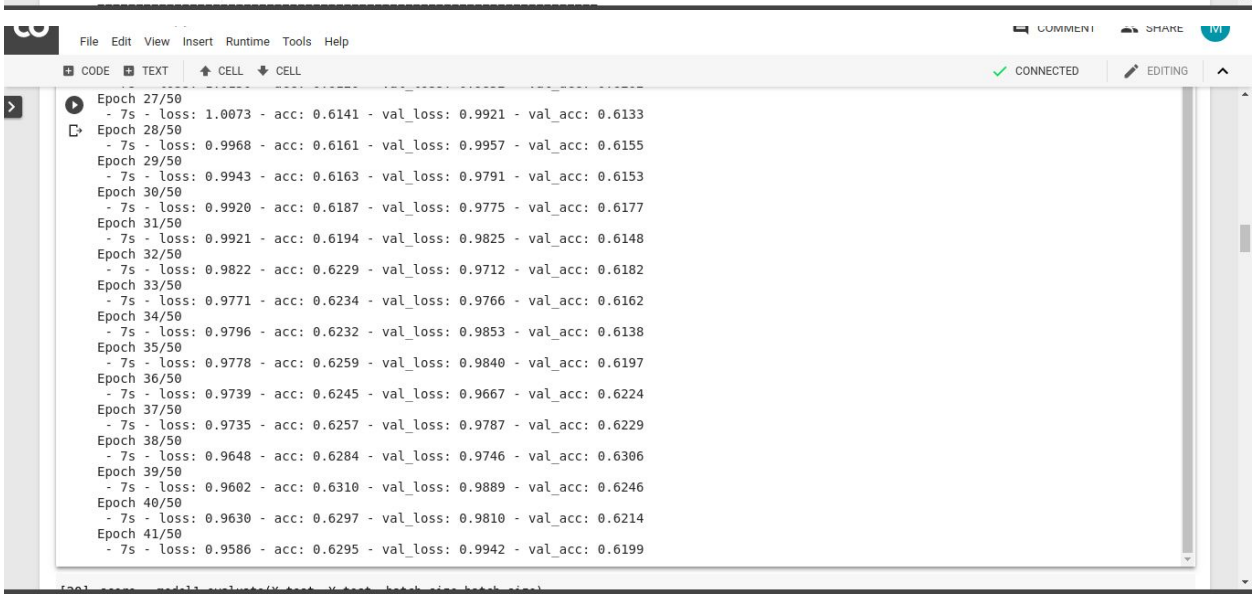

CODE
TEXT
CELL
CELL
CONNECTED
EDITING
- 7s - loss: 1.1150 - acc: 0.5683 - val_loss: 1.0537 - val_acc: 0.5911
Epoch 18/50
- 8s - loss: 1.0865 - acc: 0.5812 - val_loss: 1.0368 - val_acc: 0.5965
Epoch 19/50
- 7s - loss: 1.0663 - acc: 0.5895 - val_loss: 1.0327 - val_acc: 0.5987
Epoch 20/50
- 8s - loss: 1.0522 - acc: 0.5945 - val_loss: 1.0169 - val_acc: 0.6007
Epoch 21/50
- 8s - loss: 1.0396 - acc: 0.6006 - val_loss: 1.0249 - val_acc: 0.6081
Epoch 22/50
- 7s - loss: 1.0282 - acc: 0.6043 - val_loss: 1.0013 - val_acc: 0.6086
Epoch 23/50
- 8s - loss: 1.0225 - acc: 0.6056 - val_loss: 0.9970 - val_acc: 0.6155
Epoch 24/50
- 7s - loss: 1.0137 - acc: 0.6101 - val_loss: 1.0200 - val_acc: 0.6096
Epoch 25/50
- 7s - loss: 1.0055 - acc: 0.6118 - val_loss: 0.9914 - val_acc: 0.6143
Epoch 26/50
- 7s - loss: 1.0008 - acc: 0.6159 - val_loss: 0.9764 - val_acc: 0.6170
Epoch 27/50
- 8s - loss: 0.9906 - acc: 0.6180 - val_loss: 0.9816 - val_acc: 0.6182
Epoch 28/50
- 7s - loss: 0.9929 - acc: 0.6179 - val_loss: 1.0085 - val_acc: 0.6145
Epoch 29/50
- 7s - loss: 0.9857 - acc: 0.6209 - val_loss: 0.9932 - val_acc: 0.6167
Epoch 30/50
- 7s - loss: 0.9793 - acc: 0.6219 - val_loss: 0.9794 - val_acc: 0.6190
Epoch 31/50
- 8s - loss: 0.9796 - acc: 0.6223 - val_loss: 0.9816 - val_acc: 0.6209
[24]
score = model1.evaluate(X_test, Y_test, batch_size=batch_size)



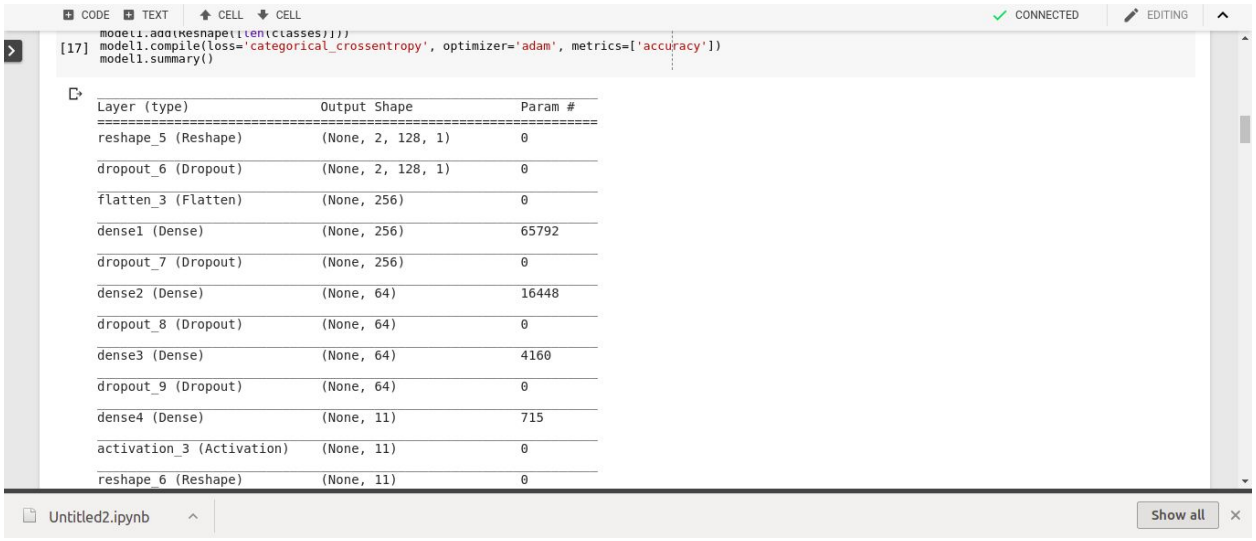
```
Layer (type)                 Output Shape              Param #
=================================================================
reshape_9 (Reshape)          (None, 2, 128, 1)         0
_________________________________________________________________
dropout_15 (Dropout)         (None, 2, 128, 1)         0
_________________________________________________________________
flatten_5 (Flatten)          (None, 256)               0
_________________________________________________________________
dense1 (Dense)               (None, 256)               65792
_________________________________________________________________
dropout_16 (Dropout)         (None, 256)               0
_________________________________________________________________
dense2 (Dense)               (None, 64)                16448
_________________________________________________________________
dropout_17 (Dropout)         (None, 64)                0
_________________________________________________________________
dense3 (Dense)               (None, 64)                4160
_________________________________________________________________
dropout_18 (Dropout)         (None, 64)                0
_________________________________________________________________
dense4 (Dense)               (None, 64)                4160
_________________________________________________________________
dropout_19 (Dropout)         (None, 64)                0
_________________________________________________________________
dense5 (Dense)               (None, 64)                4160
_________________________________________________________________
dropout_20 (Dropout)         (None, 64)                0
_________________________________________________________________
dense6 (Dense)               (None, 11)                715
_________________________________________________________________
activation_5 (Activation)    (None, 11)                0
```

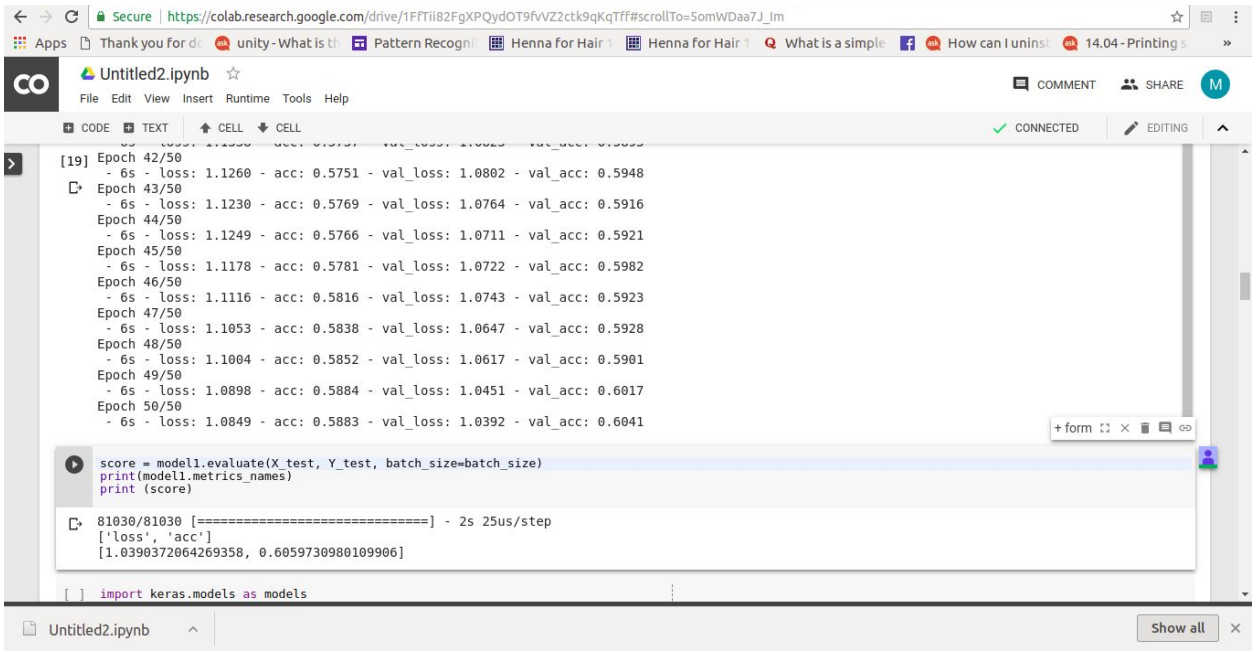


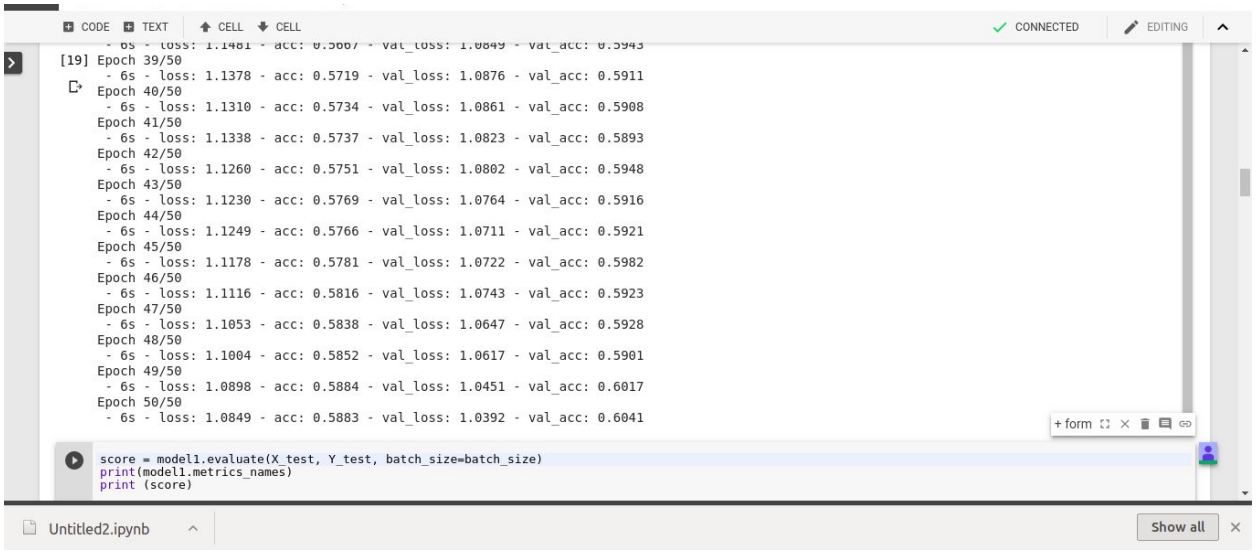
```
[21] model1.summary()

Layer (type)                 Output Shape              Param #
=================================================================
reshape_7 (Reshape)          (None, 2, 128, 1)         0
_________________________________________________________________
dropout_10 (Dropout)         (None, 2, 128, 1)         0
_________________________________________________________________
flatten_4 (Flatten)          (None, 256)               0
_________________________________________________________________
dense1 (Dense)               (None, 256)               65792
_________________________________________________________________
dropout_11 (Dropout)         (None, 256)               0
_________________________________________________________________
dense2 (Dense)               (None, 64)                16448
_________________________________________________________________
dropout_12 (Dropout)         (None, 64)                0
_________________________________________________________________
dense3 (Dense)               (None, 64)                4160
_________________________________________________________________
dropout_13 (Dropout)         (None, 64)                0
_________________________________________________________________
dense4 (Dense)               (None, 64)                4160
_________________________________________________________________
dropout_14 (Dropout)         (None, 64)                0
_________________________________________________________________
dense5 (Dense)               (None, 11)                715
_________________________________________________________________
activation_4 (Activation)    (None, 11)                0
_________________________________________________________________
reshape_8 (Reshape)          (None, 11)                0
```

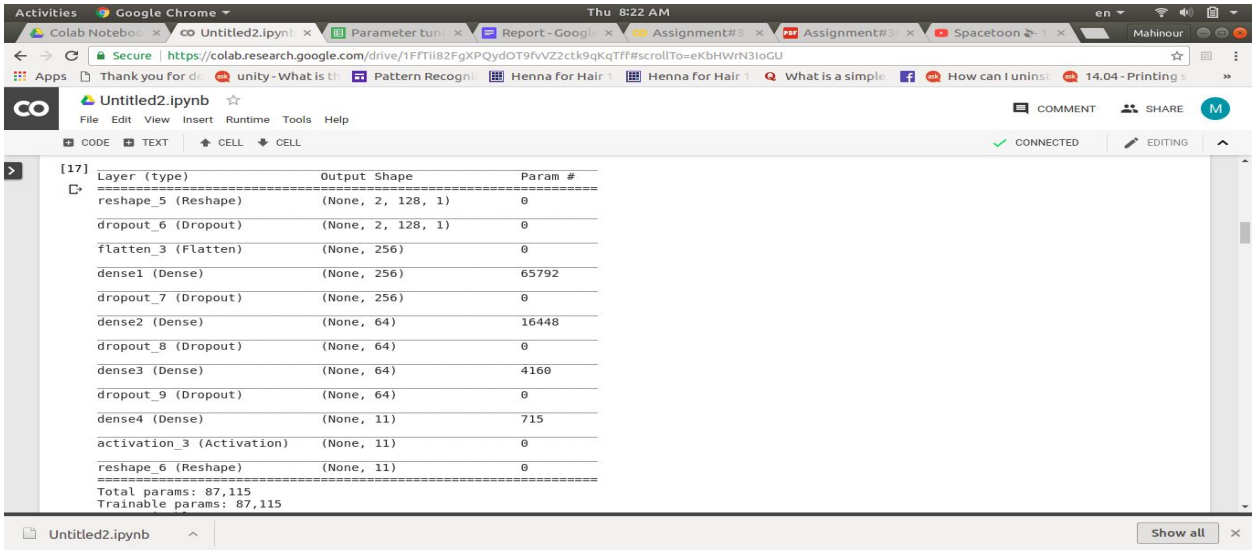


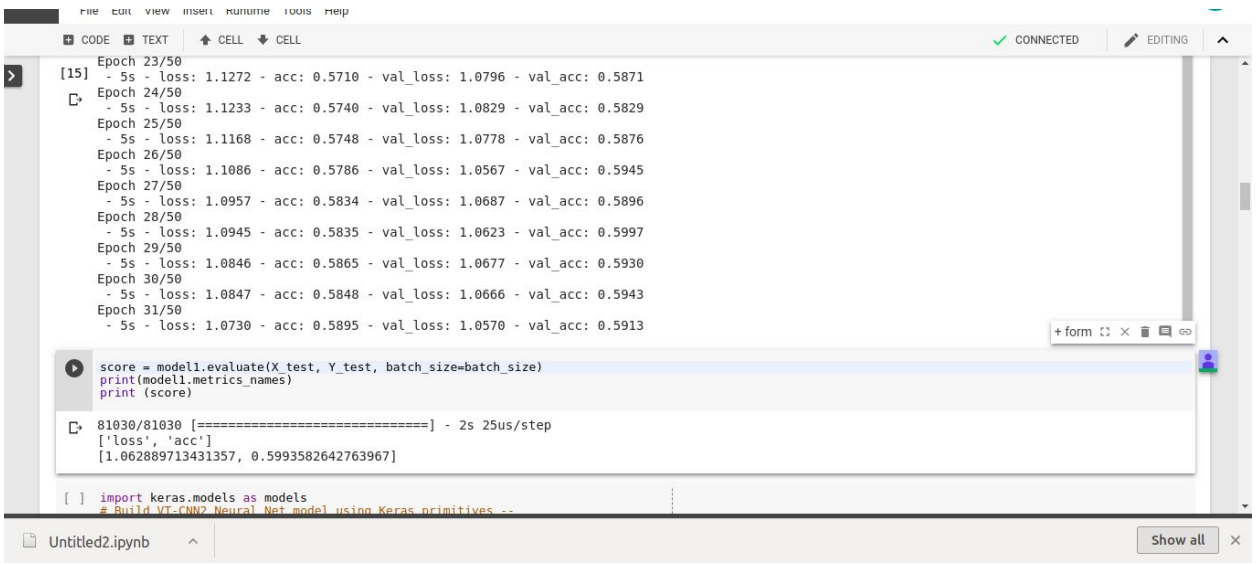
```
Epoch 27/50
 - 7s - loss: 1.0073 - acc: 0.6141 - val_loss: 0.9921 - val_acc: 0.6133
Epoch 28/50
 - 7s - loss: 0.9968 - acc: 0.6161 - val_loss: 0.9957 - val_acc: 0.6155
Epoch 29/50
 - 7s - loss: 0.9943 - acc: 0.6163 - val_loss: 0.9791 - val_acc: 0.6153
Epoch 30/50
 - 7s - loss: 0.9920 - acc: 0.6187 - val_loss: 0.9775 - val_acc: 0.6177
Epoch 31/50
 - 7s - loss: 0.9921 - acc: 0.6194 - val_loss: 0.9825 - val_acc: 0.6148
Epoch 32/50
 - 7s - loss: 0.9822 - acc: 0.6229 - val_loss: 0.9712 - val_acc: 0.6182
Epoch 33/50
 - 7s - loss: 0.9771 - acc: 0.6234 - val_loss: 0.9766 - val_acc: 0.6162
Epoch 34/50
 - 7s - loss: 0.9796 - acc: 0.6232 - val_loss: 0.9853 - val_acc: 0.6138
Epoch 35/50
 - 7s - loss: 0.9778 - acc: 0.6259 - val_loss: 0.9840 - val_acc: 0.6197
Epoch 36/50
 - 7s - loss: 0.9739 - acc: 0.6245 - val_loss: 0.9667 - val_acc: 0.6224
Epoch 37/50
 - 7s - loss: 0.9735 - acc: 0.6257 - val_loss: 0.9787 - val_acc: 0.6229
Epoch 38/50
 - 7s - loss: 0.9648 - acc: 0.6284 - val_loss: 0.9746 - val_acc: 0.6306
Epoch 39/50
 - 7s - loss: 0.9602 - acc: 0.6310 - val_loss: 0.9889 - val_acc: 0.6246
Epoch 40/50
 - 7s - loss: 0.9630 - acc: 0.6297 - val_loss: 0.9810 - val_acc: 0.6214
Epoch 41/50
 - 7s - loss: 0.9586 - acc: 0.6295 - val_loss: 0.9942 - val_acc: 0.6199
```

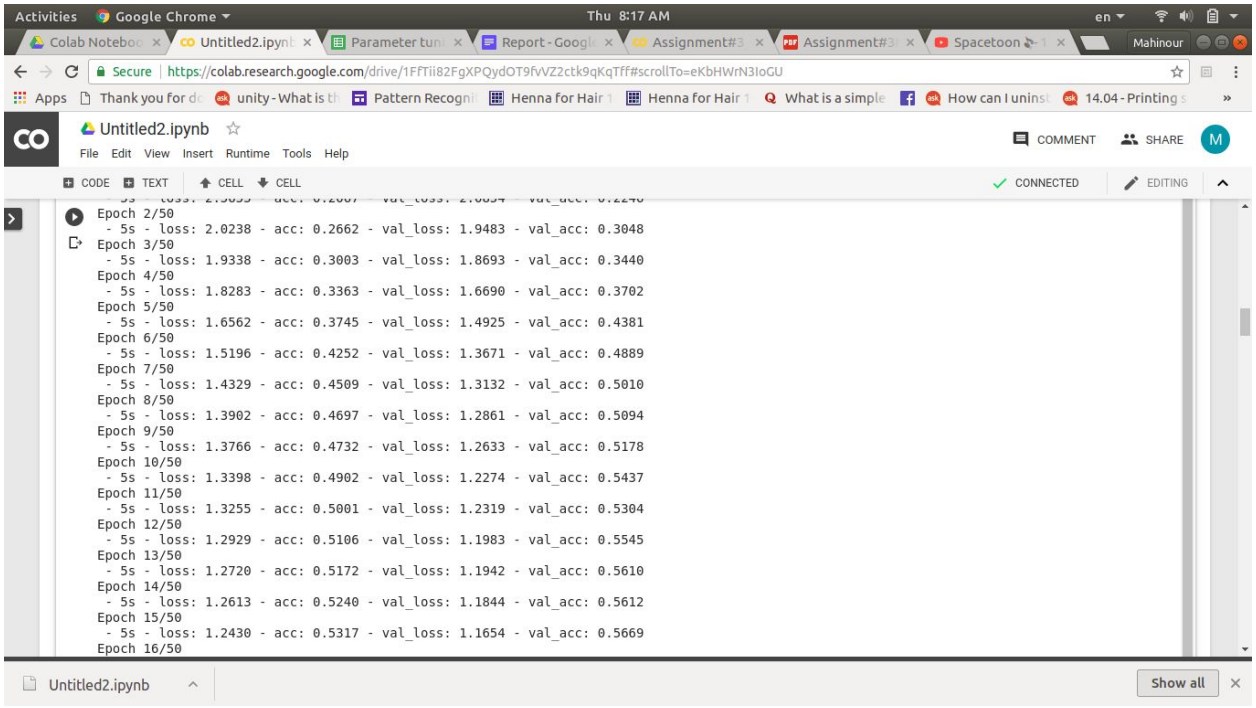

CODE TEXT CELL CELL CONNECTED EDITING
[17]
model1.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
model1.summary()
Layer (type) Output Shape Param #
reshape_5 (Reshape) (None, 2, 128, 1) 0
dropout_6 (Dropout) (None, 2, 128, 1) 0
flatten_3 (Flatten) (None, 256) 0
dense1 (Dense) (None, 256) 65792
dropout_7 (Dropout) (None, 256) 0
dense2 (Dense) (None, 64) 16448
dropout_8 (Dropout) (None, 64) 0
dense3 (Dense) (None, 64) 4160
dropout_9 (Dropout) (None, 64) 0
dense4 (Dense) (None, 11) 715
activation_3 (Activation) (None, 11) 0
reshape_6 (Reshape) (None, 11) 0
Untitled2.ipynb
Show all


Secure https://colab.research.google.com/drive/1FfTii82FgXPQydOT9fvVZ2ctk9qKqTff#scrollTo=5omWDaa7J_Im
Apps Thank you for d unity - What is th Pattern Recogni Henna for Hair Henna for Hair What is a simple How can I uninst 14.04 - Printing s
Untitled2.ipynb
File Edit View Insert Runtime Tools Help
COMMENT SHARE M
CODE TEXT CELL CELL CONNECTED EDITING
[19]
Epoch 42/50
- 6s - loss: 1.1260 - acc: 0.5751 - val_loss: 1.0802 - val_acc: 0.5948
Epoch 43/50
- 6s - loss: 1.1230 - acc: 0.5769 - val_loss: 1.0764 - val_acc: 0.5916
Epoch 44/50
- 6s - loss: 1.1249 - acc: 0.5766 - val_loss: 1.0711 - val_acc: 0.5921
Epoch 45/50
- 6s - loss: 1.1178 - acc: 0.5781 - val_loss: 1.0722 - val_acc: 0.5982
Epoch 46/50
- 6s - loss: 1.1116 - acc: 0.5816 - val_loss: 1.0743 - val_acc: 0.5923
Epoch 47/50
- 6s - loss: 1.1053 - acc: 0.5838 - val_loss: 1.0647 - val_acc: 0.5928
Epoch 48/50
- 6s - loss: 1.1004 - acc: 0.5852 - val_loss: 1.0617 - val_acc: 0.5901
Epoch 49/50
- 6s - loss: 1.0898 - acc: 0.5884 - val_loss: 1.0451 - val_acc: 0.6017
Epoch 50/50
- 6s - loss: 1.0849 - acc: 0.5883 - val_loss: 1.0392 - val_acc: 0.6041
+ form
score = model1.evaluate(X_test, Y_test, batch_size=batch_size)
print(model1.metrics_names)
print (score)
81030/81030 [==============================] - 2s 25us/step
['loss', 'acc']
[1.0390372064269358, 0.6059730980109906]
[ ] import keras.models as models
Untitled2.ipynb
Show all


CODE
TEXT
CELL
CELL
CONNECTED
EDITING
[19] Epoch 39/50
- 6s - loss: 1.1378 - acc: 0.5719 - val_loss: 1.0876 - val_acc: 0.5911
Epoch 40/50
- 6s - loss: 1.1310 - acc: 0.5734 - val_loss: 1.0861 - val_acc: 0.5908
Epoch 41/50
- 6s - loss: 1.1338 - acc: 0.5737 - val_loss: 1.0823 - val_acc: 0.5893
Epoch 42/50
- 6s - loss: 1.1260 - acc: 0.5751 - val_loss: 1.0802 - val_acc: 0.5948
Epoch 43/50
- 6s - loss: 1.1230 - acc: 0.5769 - val_loss: 1.0764 - val_acc: 0.5916
Epoch 44/50
- 6s - loss: 1.1249 - acc: 0.5766 - val_loss: 1.0711 - val_acc: 0.5921
Epoch 45/50
- 6s - loss: 1.1178 - acc: 0.5781 - val_loss: 1.0722 - val_acc: 0.5982
Epoch 46/50
- 6s - loss: 1.1116 - acc: 0.5816 - val_loss: 1.0743 - val_acc: 0.5923
Epoch 47/50
- 6s - loss: 1.1053 - acc: 0.5838 - val_loss: 1.0647 - val_acc: 0.5928
Epoch 48/50
- 6s - loss: 1.1004 - acc: 0.5852 - val_loss: 1.0617 - val_acc: 0.5901
Epoch 49/50
- 6s - loss: 1.0898 - acc: 0.5884 - val_loss: 1.0451 - val_acc: 0.6017
Epoch 50/50
- 6s - loss: 1.0849 - acc: 0.5883 - val_loss: 1.0392 - val_acc: 0.6041
+ form
score = model1.evaluate(X_test, Y_test, batch_size=batch_size)
print(model1.metrics_names)
print (score)
Untitled2.ipynb
Show all


Activities
Google Chrome
Thu 8:22 AM
en
Mahinour
Secure
https://colab.research.google.com/drive/1FfTii82FgXPQydOT9fvVZ2ctk9qKqTff#scrollTo=eKbHWrN3IoGU
Untitled2.ipynb
File Edit View Insert Runtime Tools Help
COMMENT
SHARE
CODE
TEXT
CELL
CELL
CONNECTED
EDITING
[17]
Layer (type) Output Shape Param #
reshape_5 (Reshape) (None, 2, 128, 1) 0
dropout_6 (Dropout) (None, 2, 128, 1) 0
flatten_3 (Flatten) (None, 256) 0
dense1 (Dense) (None, 256) 65792
dropout_7 (Dropout) (None, 256) 0
dense2 (Dense) (None, 64) 16448
dropout_8 (Dropout) (None, 64) 0
dense3 (Dense) (None, 64) 4160
dropout_9 (Dropout) (None, 64) 0
dense4 (Dense) (None, 11) 715
activation_3 (Activation) (None, 11) 0
reshape_6 (Reshape) (None, 11) 0
Total params: 87,115
Trainable params: 87,115
Untitled2.ipynb
Show all


File Edit View Insert Runtime Tools Help
CODE TEXT CELL CELL
CONNECTED
EDITING
[15]
Epoch 23/50
- 5s - loss: 1.1272 - acc: 0.5710 - val_loss: 1.0796 - val_acc: 0.5871
Epoch 24/50
- 5s - loss: 1.1233 - acc: 0.5740 - val_loss: 1.0829 - val_acc: 0.5829
Epoch 25/50
- 5s - loss: 1.1168 - acc: 0.5748 - val_loss: 1.0778 - val_acc: 0.5876
Epoch 26/50
- 5s - loss: 1.1086 - acc: 0.5786 - val_loss: 1.0567 - val_acc: 0.5945
Epoch 27/50
- 5s - loss: 1.0957 - acc: 0.5834 - val_loss: 1.0687 - val_acc: 0.5896
Epoch 28/50
- 5s - loss: 1.0945 - acc: 0.5835 - val_loss: 1.0623 - val_acc: 0.5997
Epoch 29/50
- 5s - loss: 1.0846 - acc: 0.5865 - val_loss: 1.0677 - val_acc: 0.5930
Epoch 30/50
- 5s - loss: 1.0847 - acc: 0.5848 - val_loss: 1.0666 - val_acc: 0.5943
Epoch 31/50
- 5s - loss: 1.0730 - acc: 0.5895 - val_loss: 1.0570 - val_acc: 0.5913
+ form
score = model1.evaluate(X_test, Y_test, batch_size=batch_size)
print(model1.metrics_names)
print (score)
81030/81030 [==============================] - 2s 25us/step
['loss', 'acc']
[1.062889713431357, 0.5993582642763967]
[ ]
import keras.models as models
# Build VT-CNN2 Neural Net model using Keras primitives --
Untitled2.ipynb
Show all


Activities
Google Chrome
Thu 8:17 AM
en
Colab Noteboo
Untitled2.ipynb
Parameter tuni
Report - Google
Assignment#3
Assignment#3
Spacetoon
Mahinour
Secure | https://colab.research.google.com/drive/1FfTii82FgXPQydOT9fvVZ2ctk9qKqTff#scrollTo=eKbHWrN3IoGU
Apps
Thank you for d
unity - What is th
Pattern Recogni
Henna for Hair
Henna for Hair
What is a simple
How can I uninst
14.04 - Printing s
Untitled2.ipynb
File Edit View Insert Runtime Tools Help
COMMENT
SHARE
M
CODE TEXT CELL CELL
CONNECTED
EDITING
Epoch 2/50
- 5s - loss: 2.0238 - acc: 0.2662 - val_loss: 1.9483 - val_acc: 0.3048
Epoch 3/50
- 5s - loss: 1.9338 - acc: 0.3003 - val_loss: 1.8693 - val_acc: 0.3440
Epoch 4/50
- 5s - loss: 1.8283 - acc: 0.3363 - val_loss: 1.6690 - val_acc: 0.3702
Epoch 5/50
- 5s - loss: 1.6562 - acc: 0.3745 - val_loss: 1.4925 - val_acc: 0.4381
Epoch 6/50
- 5s - loss: 1.5196 - acc: 0.4252 - val_loss: 1.3671 - val_acc: 0.4889
Epoch 7/50
- 5s - loss: 1.4329 - acc: 0.4509 - val_loss: 1.3132 - val_acc: 0.5010
Epoch 8/50
- 5s - loss: 1.3902 - acc: 0.4697 - val_loss: 1.2861 - val_acc: 0.5094
Epoch 9/50
- 5s - loss: 1.3766 - acc: 0.4732 - val_loss: 1.2633 - val_acc: 0.5178
Epoch 10/50
- 5s - loss: 1.3398 - acc: 0.4902 - val_loss: 1.2274 - val_acc: 0.5437
Epoch 11/50
- 5s - loss: 1.3255 - acc: 0.5001 - val_loss: 1.2319 - val_acc: 0.5304
Epoch 12/50
- 5s - loss: 1.2929 - acc: 0.5106 - val_loss: 1.1983 - val_acc: 0.5545
Epoch 13/50
- 5s - loss: 1.2720 - acc: 0.5172 - val_loss: 1.1942 - val_acc: 0.5610
Epoch 14/50
- 5s - loss: 1.2613 - acc: 0.5240 - val_loss: 1.1844 - val_acc: 0.5612
Epoch 15/50
- 5s - loss: 1.2430 - acc: 0.5317 - val_loss: 1.1654 - val_acc: 0.5669
Epoch 16/50
Untitled2.ipynb
Show all

CODE TEXT CELL CELL CONNECTED EDITING

[13]

```
Layer (type)                 Output Shape              Param #
=================================================================
reshape_3 (Reshape)          (None, 2, 128, 1)         0
_________________________________________________________________
dropout_3 (Dropout)          (None, 2, 128, 1)         0
_________________________________________________________________
flatten_2 (Flatten)          (None, 256)               0
_________________________________________________________________
dense1 (Dense)               (None, 256)               65792
_________________________________________________________________
dropout_4 (Dropout)          (None, 256)               0
_________________________________________________________________
dense2 (Dense)               (None, 64)                16448
_________________________________________________________________
dropout_5 (Dropout)          (None, 64)                0
_________________________________________________________________
dense4 (Dense)               (None, 11)                715
_________________________________________________________________
activation_2 (Activation)    (None, 11)                0
_________________________________________________________________
reshape_4 (Reshape)          (None, 11)                0
=================================================================
Total params: 82,955
Trainable params: 82,955
Non-trainable params: 0
_________________________________________________________________
```

Untitled2.ipynb Show all

CODE TEXT CELL CELL CONNECTED EDITING

[13]

```
reshape_4 (Reshape)          (None, 11)                0
=================================================================
Total params: 82,955
Trainable params: 82,955
Non-trainable params: 0
_________________________________________________________________
```

[14]

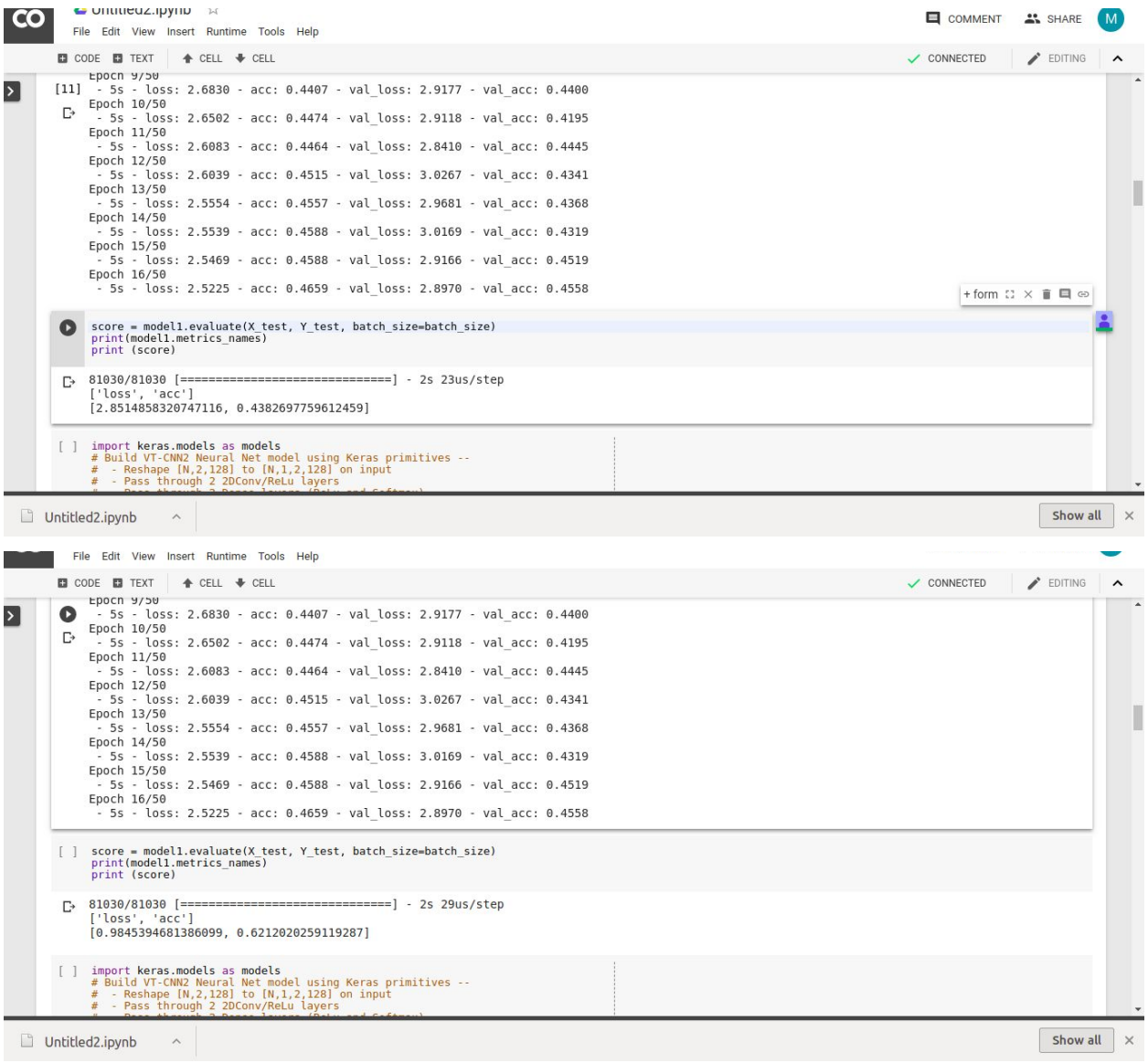
```
# Set up some params
nb_epoch = 50   # number of epochs to train on
batch_size = 100  # training batch size
```

+ form

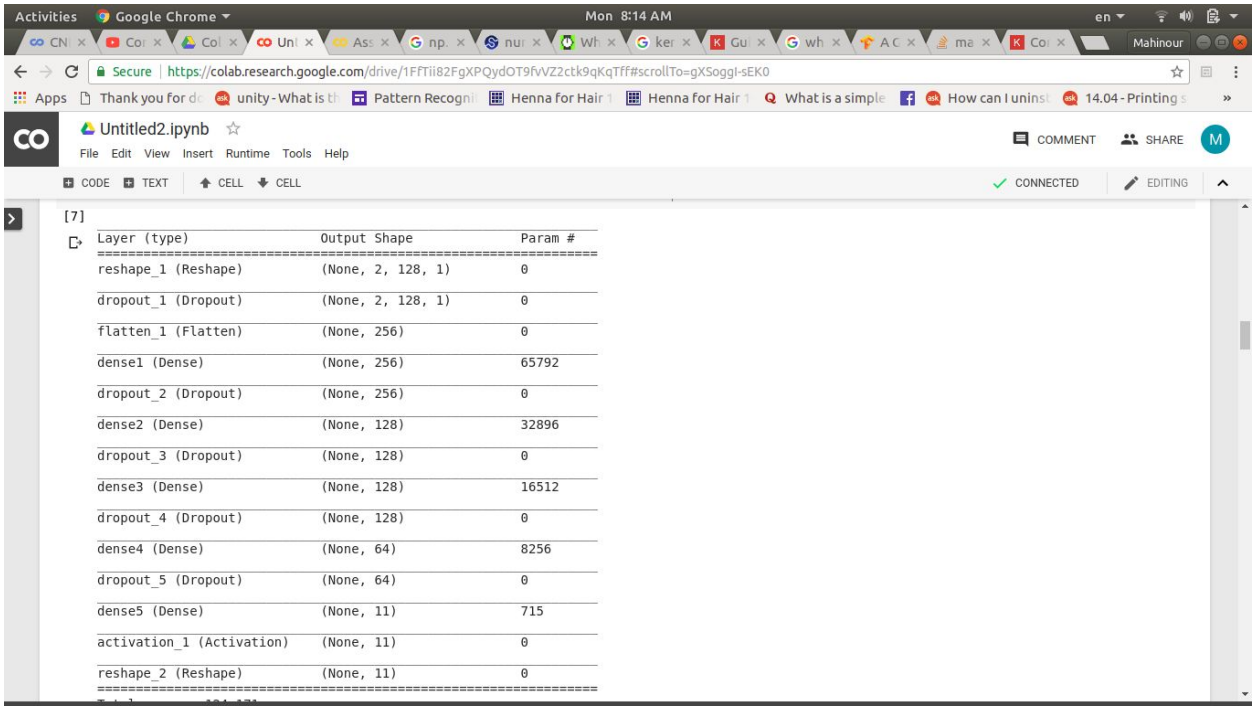
```
# perform training ...
#   - call the main training loop in keras for our network+dataset
filepath = 'convmodrecnets_CNN2_0.5.wts.h5'
history=model1.fit(X_train,
    Y_train,
    batch_size=batch_size,
    epochs=nb_epoch,
    verbose=2,
     validation_split=0.05,
    callbacks = [
        keras.callbacks.ModelCheckpoint(filepath, monitor='val_loss', verbose=0, save_best_only=True, mode='auto'),
        keras.callbacks.EarlyStopping(monitor='val_loss', patience=5, verbose=0, mode='auto')
    ])
# we re-load the best weights once training is finished
model1.load_weights(filepath)
```

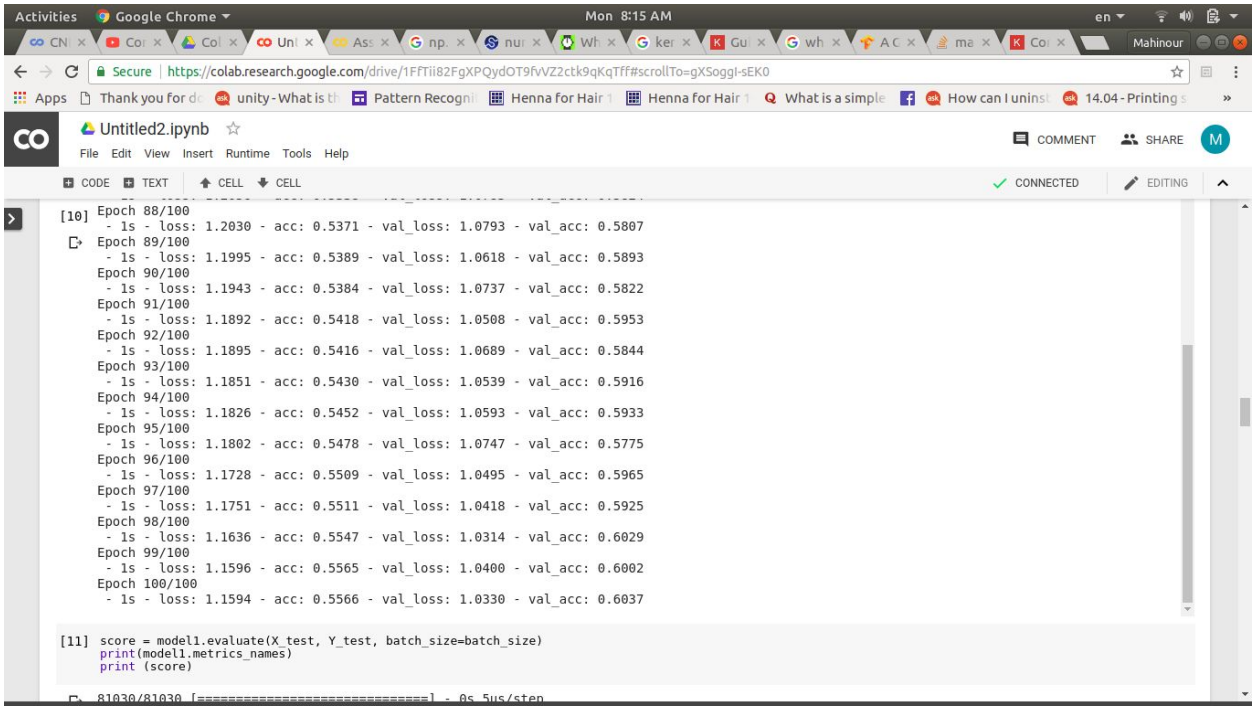
```
Train on 76978 samples, validate on 4052 samples
Epoch 1/50
 - 5s - loss: 2.5655 - acc: 0.2067 - val_loss: 2.0854 - val_acc: 0.2246
```

Untitled2.ipynb Show all


Untitled2.ipynb
File Edit View Insert Runtime Tools Help
COMMENT
SHARE
CODE
TEXT
CELL
CELL
CONNECTED
EDITING
Epoch 9/50
[11] - 5s - loss: 2.6830 - acc: 0.4407 - val_loss: 2.9177 - val_acc: 0.4400
Epoch 10/50
- 5s - loss: 2.6502 - acc: 0.4474 - val_loss: 2.9118 - val_acc: 0.4195
Epoch 11/50
- 5s - loss: 2.6083 - acc: 0.4464 - val_loss: 2.8410 - val_acc: 0.4445
Epoch 12/50
- 5s - loss: 2.6039 - acc: 0.4515 - val_loss: 3.0267 - val_acc: 0.4341
Epoch 13/50
- 5s - loss: 2.5554 - acc: 0.4557 - val_loss: 2.9681 - val_acc: 0.4368
Epoch 14/50
- 5s - loss: 2.5539 - acc: 0.4588 - val_loss: 3.0169 - val_acc: 0.4319
Epoch 15/50
- 5s - loss: 2.5469 - acc: 0.4588 - val_loss: 2.9166 - val_acc: 0.4519
Epoch 16/50
- 5s - loss: 2.5225 - acc: 0.4659 - val_loss: 2.8970 - val_acc: 0.4558
+ form
score = model1.evaluate(X_test, Y_test, batch_size=batch_size)
print(model1.metrics_names)
print (score)
81030/81030 [==============================] - 2s 23us/step
['loss', 'acc']
[2.8514858320747116, 0.4382697759612459]
[ ] import keras.models as models
Build VT-CNN2 Neural Net model using Keras primitives --
- Reshape [N,2,128] to [N,1,2,128] on input
- Pass through 2 2DConv/ReLu layers
Untitled2.ipynb
Show all

File Edit View Insert Runtime Tools Help
CODE
TEXT
CELL
CELL
CONNECTED
EDITING
Epoch 9/50
- 5s - loss: 2.6830 - acc: 0.4407 - val_loss: 2.9177 - val_acc: 0.4400
Epoch 10/50
- 5s - loss: 2.6502 - acc: 0.4474 - val_loss: 2.9118 - val_acc: 0.4195
Epoch 11/50
- 5s - loss: 2.6083 - acc: 0.4464 - val_loss: 2.8410 - val_acc: 0.4445
Epoch 12/50
- 5s - loss: 2.6039 - acc: 0.4515 - val_loss: 3.0267 - val_acc: 0.4341
Epoch 13/50
- 5s - loss: 2.5554 - acc: 0.4557 - val_loss: 2.9681 - val_acc: 0.4368
Epoch 14/50
- 5s - loss: 2.5539 - acc: 0.4588 - val_loss: 3.0169 - val_acc: 0.4319
Epoch 15/50
- 5s - loss: 2.5469 - acc: 0.4588 - val_loss: 2.9166 - val_acc: 0.4519
Epoch 16/50
- 5s - loss: 2.5225 - acc: 0.4659 - val_loss: 2.8970 - val_acc: 0.4558
[ ] score = model1.evaluate(X_test, Y_test, batch_size=batch_size)
print(model1.metrics_names)
print (score)
81030/81030 [==============================] - 2s 29us/step
['loss', 'acc']
[0.9845394681386099, 0.6212020259119287]
[ ] import keras.models as models
Build VT-CNN2 Neural Net model using Keras primitives --
- Reshape [N,2,128] to [N,1,2,128] on input
- Pass through 2 2DConv/ReLu layers
Untitled2.ipynb
Show all


Activities
Google Chrome
Mon 8:14 AM
en
Mahinour
Secure | https://colab.research.google.com/drive/1FfTii82FgXPQydOT9fvVZ2ctk9qKqTff#scrollTo=gXSoggI-sEK0
Untitled2.ipynb
File Edit View Insert Runtime Tools Help
COMMENT
SHARE
CODE TEXT CELL CELL
CONNECTED
EDITING
[7]
Layer (type) Output Shape Param #
reshape_1 (Reshape) (None, 2, 128, 1) 0
dropout_1 (Dropout) (None, 2, 128, 1) 0
flatten_1 (Flatten) (None, 256) 0
dense1 (Dense) (None, 256) 65792
dropout_2 (Dropout) (None, 256) 0
dense2 (Dense) (None, 128) 32896
dropout_3 (Dropout) (None, 128) 0
dense3 (Dense) (None, 128) 16512
dropout_4 (Dropout) (None, 128) 0
dense4 (Dense) (None, 64) 8256
dropout_5 (Dropout) (None, 64) 0
dense5 (Dense) (None, 11) 715
activation_1 (Activation) (None, 11) 0
reshape_2 (Reshape) (None, 11) 0


Activities
Google Chrome
Mon 8:15 AM
en
Mahinour
Secure | https://colab.research.google.com/drive/1FfTii82FgXPQydOT9fvVZ2ctk9qKqTff#scrollTo=gXSoggI-sEK0
Untitled2.ipynb
File Edit View Insert Runtime Tools Help
COMMENT
SHARE
CODE TEXT CELL CELL
CONNECTED
EDITING
[10] Epoch 88/100
- 1s - loss: 1.2030 - acc: 0.5371 - val_loss: 1.0793 - val_acc: 0.5807
Epoch 89/100
- 1s - loss: 1.1995 - acc: 0.5389 - val_loss: 1.0618 - val_acc: 0.5893
Epoch 90/100
- 1s - loss: 1.1943 - acc: 0.5384 - val_loss: 1.0737 - val_acc: 0.5822
Epoch 91/100
- 1s - loss: 1.1892 - acc: 0.5418 - val_loss: 1.0508 - val_acc: 0.5953
Epoch 92/100
- 1s - loss: 1.1895 - acc: 0.5416 - val_loss: 1.0689 - val_acc: 0.5844
Epoch 93/100
- 1s - loss: 1.1851 - acc: 0.5430 - val_loss: 1.0539 - val_acc: 0.5916
Epoch 94/100
- 1s - loss: 1.1826 - acc: 0.5452 - val_loss: 1.0593 - val_acc: 0.5933
Epoch 95/100
- 1s - loss: 1.1802 - acc: 0.5478 - val_loss: 1.0747 - val_acc: 0.5775
Epoch 96/100
- 1s - loss: 1.1728 - acc: 0.5509 - val_loss: 1.0495 - val_acc: 0.5965
Epoch 97/100
- 1s - loss: 1.1751 - acc: 0.5511 - val_loss: 1.0418 - val_acc: 0.5925
Epoch 98/100
- 1s - loss: 1.1636 - acc: 0.5547 - val_loss: 1.0314 - val_acc: 0.6029
Epoch 99/100
- 1s - loss: 1.1596 - acc: 0.5565 - val_loss: 1.0400 - val_acc: 0.6002
Epoch 100/100
- 1s - loss: 1.1594 - acc: 0.5566 - val_loss: 1.0330 - val_acc: 0.6037
[11] score = model1.evaluate(X_test, Y_test, batch_size=batch_size)
print(model1.metrics_names)
print (score)
81030/81030 [==========] - 0s 5us/step

Activities Google Chrome Mon 8:15 AM en

Secure https://colab.research.google.com/drive/1FfTii82FgXPQydOT9fvVZ2ctk9qKqTff#scrollTo=gXSoggI-sEK0

Untitled2.ipynb

File Edit View Insert Runtime Tools Help

COMMENT SHARE

CODE TEXT CELL CELL CONNECTED EDITING

```
[10] Epoch 91/100
 - 1s - loss: 1.1892 - acc: 0.5418 - val_loss: 1.0508 - val_acc: 0.5953
Epoch 92/100
 - 1s - loss: 1.1895 - acc: 0.5416 - val_loss: 1.0689 - val_acc: 0.5844
Epoch 93/100
 - 1s - loss: 1.1851 - acc: 0.5430 - val_loss: 1.0539 - val_acc: 0.5916
Epoch 94/100
 - 1s - loss: 1.1826 - acc: 0.5452 - val_loss: 1.0593 - val_acc: 0.5933
Epoch 95/100
 - 1s - loss: 1.1802 - acc: 0.5478 - val_loss: 1.0747 - val_acc: 0.5775
Epoch 96/100
 - 1s - loss: 1.1728 - acc: 0.5509 - val_loss: 1.0495 - val_acc: 0.5965
Epoch 97/100
 - 1s - loss: 1.1751 - acc: 0.5511 - val_loss: 1.0418 - val_acc: 0.5925
Epoch 98/100
 - 1s - loss: 1.1636 - acc: 0.5547 - val_loss: 1.0314 - val_acc: 0.6029
Epoch 99/100
 - 1s - loss: 1.1596 - acc: 0.5565 - val_loss: 1.0400 - val_acc: 0.6002
Epoch 100/100
 - 1s - loss: 1.1594 - acc: 0.5566 - val_loss: 1.0330 - val_acc: 0.6037
```

```
[11] score = model1.evaluate(X_test, Y_test, batch_size=batch_size)
     print(model1.metrics_names)
     print (score)
```

```
81030/81030 [==============================] - 0s 5us/step
['loss', 'acc']
[1.017300804427571, 0.6019005308233571]
```

```
[ ] import keras.models as models
    # Build VT-CNN2 Neural Net model using Keras primitives --
    #  - Reshape [N,2,128] to [N,1,2,128] on input
```

Activities Google Chrome Mon 8:15 AM en

Secure https://colab.research.google.com/drive/1FfTii82FgXPQydOT9fvVZ2ctk9qKqTff#scrollTo=gXSoggI-sEK0

Untitled2.ipynb

File Edit View Insert Runtime Tools Help

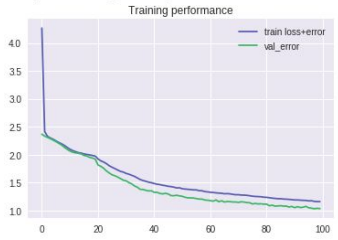
COMMENT SHARE

CODE TEXT CELL CELL CONNECTED EDITING

```
[1.0656827691888204, 0.6100826851934411]
```

```
[12] # Show loss curves
     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()
```

```
<matplotlib.legend.Legend at 0x7feba46da470>
```




```
[13] 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)
```

Activities Google Chrome Mon 8:15 AM en

Counting Stars
OneRepublic - Native

Secure | https://colab.research.google.com/drive

Apps Thank you for d unity - What is th Pattern Recogni Henna for Hair Henna for Hair What is a simple How can I uninst 14.04 - Printing s

Untitled2.ipynb

File Edit View Insert Runtime Tools Help

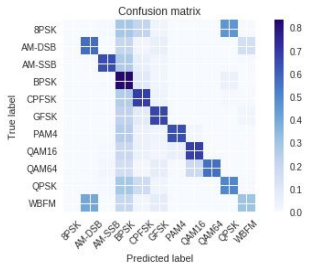
COMMENT SHARE

CODE TEXT CELL CELL CONNECTED EDITING

```
[13]    plt.xlabel('Predicted label')
```

+ form

```
# Plot confusion matrix
test_Y_hat = model1.predict(X_test, batch_size=batch_size)
conf = np.zeros([len(classes),len(classes)])
confnorm = np.zeros([len(classes),len(classes)])
for i in range(0,X_test.shape[0]):
    j = list(Y_test[i,:]).index(1)
    k = int(np.argmax(test_Y_hat[i,:]))
    conf[j,k] = conf[j,k] + 1
for i in range(0,len(classes)):
    confnorm[i,:] = conf[i,:] / np.sum(conf[i,:])
plot_confusion_matrix(confnorm, labels=classes)
# on the right its a color chart to indicate that the darker the color the easier and thee most to identify
```



Activities Google Chrome Mon 8:16 AM en

Secure | https://colab.research.google.com/drive/1FfTii82FgXPQydOT9fvVZ2ctk9qKqTff#scrollTo=2l9abc-FsDdE

Apps Thank you for d unity - What is th Pattern Recogni Henna for Hair Henna for Hair What is a simple How can I uninst 14.04 - Printing s

Untitled2.ipynb

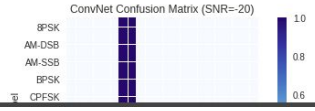
File Edit View Insert Runtime Tools Help

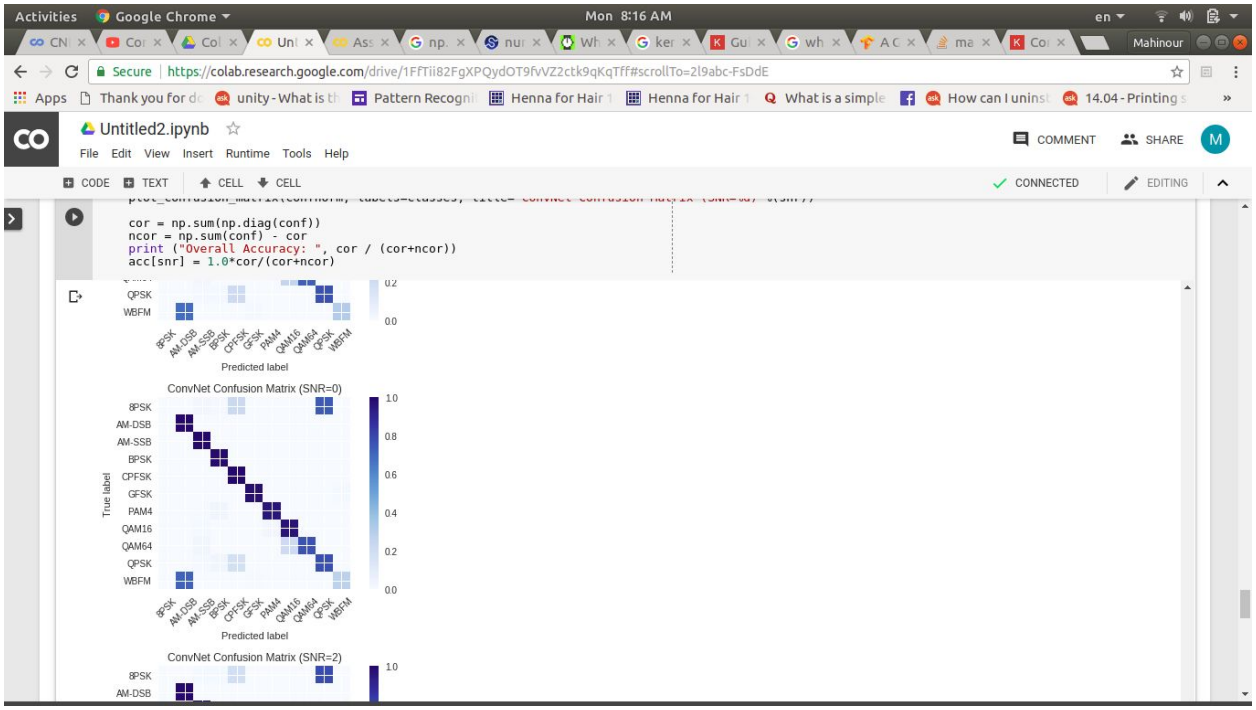
COMMENT SHARE

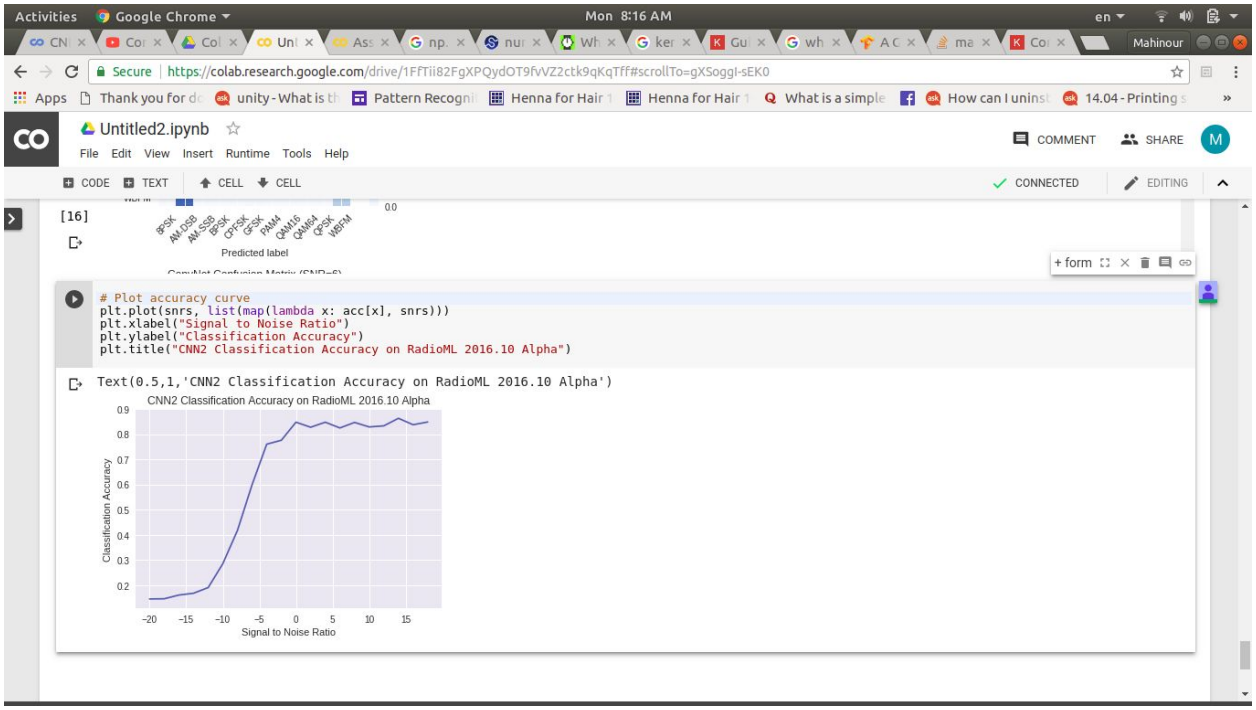
CODE TEXT CELL CELL CONNECTED EDITING

```
    cor = np.sum(np.diag(conf))
    ncor = np.sum(conf) - cor
    print ("Overall Accuracy: ", cor / (cor+ncor))
    acc[snr] = 1.0*cor/(cor+ncor)
```

```
Overall Accuracy:  0.14537335648722402
Overall Accuracy:  0.14622757434259032
Overall Accuracy:  0.16099605522682445
Overall Accuracy:  0.16819875776397517
Overall Accuracy:  0.19061441702951137
Overall Accuracy:  0.28546840422916153
Overall Accuracy:  0.41758510898848883
Overall Accuracy:  0.5992102665350444
Overall Accuracy:  0.7598991172761664
Overall Accuracy:  0.7755856966707768
Overall Accuracy:  0.8473566641846612
Overall Accuracy:  0.827203159713651
Overall Accuracy:  0.847239114570664
Overall Accuracy:  0.8244575936883629
Overall Accuracy:  0.8463426607756505
Overall Accuracy:  0.8284797630799605
Overall Accuracy:  0.8327543424317618
Overall Accuracy:  0.8627646326276464
Overall Accuracy:  0.8371520078837152
Overall Accuracy:  0.8481075697211156
```




Activities
Google Chrome
Mon 8:16 AM
en
Mahinour
Secure | https://colab.research.google.com/drive/1FfTii82FgXPQydOT9fvVZ2ctk9qKqTff#scrollTo=2l9abc-FsDdE
Untitled2.ipynb
File Edit View Insert Runtime Tools Help
COMMENT
SHARE
CODE
TEXT
CELL
CELL
CONNECTED
EDITING
cor = np.sum(np.diag(conf))
ncor = np.sum(conf) - cor
print ("Overall Accuracy: ", cor / (cor+ncor))
acc[snr] = 1.0*cor/(cor+ncor)
QPSK
WBFM
8PSK
AM-DSB
AM-SSB
BPSK
CPFSK
GFSK
PAM4
QAM16
QAM64
QPSK
WBFM
Predicted label
ConvNet Confusion Matrix (SNR=0)
True label
ConvNet Confusion Matrix (SNR=2)


Activities
Google Chrome
Mon 8:16 AM
en
Mahinour
Secure | https://colab.research.google.com/drive/1FfTii82FgXPQydOT9fvVZ2ctk9qKqTff#scrollTo=gXSoggI-sEK0
Untitled2.ipynb
File Edit View Insert Runtime Tools Help
COMMENT
SHARE
CODE
TEXT
CELL
CELL
CONNECTED
EDITING
[16]
Predicted label
+ form
# Plot accuracy curve
plt.plot(snrs, list(map(lambda x: acc[x], snrs)))
plt.xlabel("Signal to Noise Ratio")
plt.ylabel("Classification Accuracy")
plt.title("CNN2 Classification Accuracy on RadioML 2016.10 Alpha")
Text(0.5,1,'CNN2 Classification Accuracy on RadioML 2016.10 Alpha')
CNN2 Classification Accuracy on RadioML 2016.10 Alpha
Classification Accuracy
Signal to Noise Ratio

Activities Google Chrome Mon 8:30 AM en Mahinour

Secure | https://colab.research.google.com/drive/1FfTii82FgXPQydOT9fvVZ2ctk9qKqTff#scrollTo=gXSoggI-sEK0

Apps Thank you for d unity - What is th Pattern Recogni Henna for Hair Henna for Hair What is a simple How can I uninst 14.04 - Printing s

Untitled2.ipynb

File Edit View Insert Runtime Tools Help

COMMENT SHARE M

CODE TEXT CELL CELL CONNECTED EDITING

```
[20]  Epoch 51/100
       - 1s - loss: 0.9055 - acc: 0.6479 - val_loss: 0.9602 - val_acc: 0.6276
      Epoch 52/100
       - 1s - loss: 0.9003 - acc: 0.6484 - val_loss: 0.9566 - val_acc: 0.6288
      Epoch 53/100
       - 1s - loss: 0.8972 - acc: 0.6486 - val_loss: 0.9529 - val_acc: 0.6347
      Epoch 54/100
       - 1s - loss: 0.8963 - acc: 0.6510 - val_loss: 0.9722 - val_acc: 0.6325
      Epoch 55/100
       - 1s - loss: 0.8885 - acc: 0.6555 - val_loss: 0.9617 - val_acc: 0.6399
      Epoch 56/100
       - 1s - loss: 0.8854 - acc: 0.6534 - val_loss: 0.9718 - val_acc: 0.6338
      Epoch 57/100
       - 1s - loss: 0.8820 - acc: 0.6565 - val_loss: 0.9528 - val_acc: 0.6365
      Epoch 58/100
       - 1s - loss: 0.8817 - acc: 0.6570 - val_loss: 0.9549 - val_acc: 0.6382
      Epoch 59/100
       - 1s - loss: 0.8792 - acc: 0.6567 - val_loss: 0.9625 - val_acc: 0.6343
      Epoch 60/100
       - 1s - loss: 0.8763 - acc: 0.6592 - val_loss: 0.9654 - val_acc: 0.6340
      Epoch 61/100
       - 1s - loss: 0.8747 - acc: 0.6594 - val_loss: 0.9592 - val_acc: 0.6325
      Epoch 62/100
       - 1s - loss: 0.8678 - acc: 0.6619 - val_loss: 0.9710 - val_acc: 0.6355
```

```
[21]  score = model1.evaluate(X_test, Y_test, batch_size=batch_size)
      print(model1.metrics_names)
      print (score)
```

```
      81030/81030 [==============================] - 0s 5us/step
      ['loss', 'acc']
      [0.9580647699782234, 0.6355423914627442]
```

Activities Google Chrome Mon 8:30 AM en Mahinour

Secure | https://colab.research.google.com/drive/1FfTii82FgXPQydOT9fvVZ2ctk9qKqTff#scrollTo=gXSoggI-sEK0

Apps Thank you for d unity - What is th Pattern Recogni Henna for Hair Henna for Hair What is a simple How can I uninst 14.04 - Printing s

Untitled2.ipynb

File Edit View Insert Runtime Tools Help

COMMENT SHARE M

CODE TEXT CELL CELL CONNECTED EDITING

```
[22]  # Show loss curves
      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()
```

```
      <matplotlib.legend.Legend at 0x7feb98fedd68>
```


```
[23]  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')
```

Activities Google Chrome Mon 8:30 AM en Mahinour

Secure https://colab.research.google.com/drive/1FfTii82FgXPQydOT9fvVZ2ctk9qKqTff#scrollTo=gXSoggI-sEK0

Apps Thank you for d unity - What is th Pattern Recogni Henna for Hair Henna for Hair What is a simple How can I uninst 14.04 - Printing s

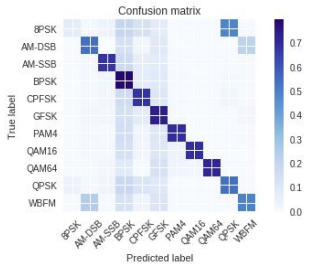
Untitled2.ipynb

File Edit View Insert Runtime Tools Help

COMMENT SHARE M

CODE TEXT CELL CELL CONNECTED EDITING

```
[24] # Plot confusion matrix
     test_Y_hat = model1.predict(X_test, batch_size=batch_size)
     conf = np.zeros([len(classes),len(classes)])
     confnorm = np.zeros([len(classes),len(classes)])
     for i in range(0,X_test.shape[0]):
         j = list(Y_test[i,:]).index(1)
         k = int(np.argmax(test_Y_hat[i,:]))
         conf[j,k] = conf[j,k] + 1
     for i in range(0,len(classes)):
         confnorm[i,:] = conf[i,:] / np.sum(conf[i,:])
     plot_confusion_matrix(confnorm, labels=classes)
     # on the right its a color chart to indicate that the darker the color the easier and thee most to identify
```




```
[25] # Plot confusion matrix
     acc = {}
```

Activities Google Chrome Mon 8:30 AM en Mahinour

Secure https://colab.research.google.com/drive/1FfTii82FgXPQydOT9fvVZ2ctk9qKqTff#scrollTo=gXSoggI-sEK0

Apps Thank you for d unity - What is th Pattern Recogni Henna for Hair Henna for Hair What is a simple How can I uninst 14.04 - Printing s

Untitled2.ipynb

File Edit View Insert Runtime Tools Help

COMMENT SHARE M

CODE TEXT CELL CELL CONNECTED EDITING

```
[25]         conf[j,k] = conf[j,k] + 1
     for i in range(0,len(classes)):
         confnorm[i,:] = conf[i,:] / np.sum(conf[i,:])
     plt.figure()
     plot_confusion_matrix(confnorm, labels=classes, title="ConvNet Confusion Matrix (SNR=%d)"%(snr))

     cor = np.sum(np.diag(conf))
     ncor = np.sum(conf) - cor
     print ("Overall Accuracy: ", cor / (cor+ncor))
     acc[snr] = 1.0*cor/(cor+ncor)
```

```
Overall Accuracy:  0.1538079880922848
Overall Accuracy:  0.14966822315065126
Overall Accuracy:  0.16247534516765286
Overall Accuracy:  0.1759006211180124
Overall Accuracy:  0.21504596032897919
Overall Accuracy:  0.2776001967051881
Overall Accuracy:  0.4785696791574822
Overall Accuracy:  0.6372161895360315
Overall Accuracy:  0.7702395964691047
Overall Accuracy:  0.8589395807644883
Overall Accuracy:  0.8602630925788036
Overall Accuracy:  0.8457171068871884
Overall Accuracy:  0.8805643395767453
Overall Accuracy:  0.9154339250493096
Overall Accuracy:  0.8465881197839961
Overall Accuracy:  0.9010365251727542
Overall Accuracy:  0.9255583126550868
Overall Accuracy:  0.8528019925280199
Overall Accuracy:  0.887410692288741
Overall Accuracy:  0.9287848605577689
```

Activities Google Chrome Mon 8:30 AM en Mahinour

Secure | https://colab.research.google.com/drive/1FfTii82FgXPQydOT9fvVZ2ctk9qKqTff#scrollTo=gXSoggI-sEK0

Apps Thank you for d unity - What is th Pattern Recogni Henna for Hair Henna for Hair What is a simple How can I uninst 14.04 - Printing s

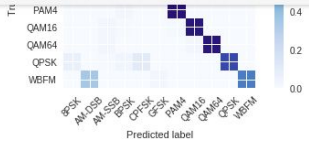
Untitled2.ipynb

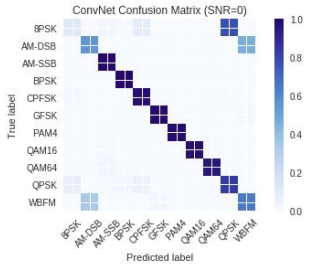
File Edit View Insert Runtime Tools Help

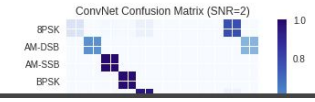
COMMENT SHARE M

CODE TEXT CELL CELL CONNECTED EDITING

[25]







Activities Google Chrome Mon 8:30 AM en Mahinour

Secure | https://colab.research.google.com/drive/1FfTii82FgXPQydOT9fvVZ2ctk9qKqTff#scrollTo=gXSoggI-sEK0

Apps Thank you for d unity - What is th Pattern Recogni Henna for Hair Henna for Hair What is a simple How can I uninst 14.04 - Printing s

Untitled2.ipynb

File Edit View Insert Runtime Tools Help

COMMENT SHARE M

CODE TEXT CELL CELL CONNECTED EDITING

+ form

```
# Plot accuracy curve
plt.plot(snrs, list(map(lambda x: acc[x], snrs)))
plt.xlabel("Signal to Noise Ratio")
plt.ylabel("Classification Accuracy")
plt.title("CNN2 Classification Accuracy on RadioML 2016.10 Alpha")
```

```
Text(0.5,1,'CNN2 Classification Accuracy on RadioML 2016.10 Alpha')
```