CS6611- CREATIVE AND INNOVATIVE PROJECT

B.E CSE VI Q - BATCH

TEAM MEMBERS: **TEAM NO: 12**

S.NO.	REG. NO.	NAME
1.	2018103569	Niranjan K
2.	2018103063	Shankar Kumar S
3.	2018103620	Vedanth S

Project Title: Identifying Sound Labels from Spectrograms using Deep Neural Networks

WEEK 11 - EXECUTION (90% IMPLEMENTATION)

Observation document (5)	
On the Spot exercise (5)	
Laboratory exercises identified (15)	
Total (25)	

TRAINING A MODEL ON MFCC-SPECTROGRAM

In [5]:
```
x_train_mfcc = x_train_mfcc[1:,:,:]
```

In [6]:
```
x_train_mfcc = x_train_mfcc[:,:,:,np.newaxis]
```

In [7]:
```
y_train_mfcc = []
for i in range(3799):
    y_train_mfcc.append(train_mfcc_spec_normalised[i][1])
```

In [8]:
```
y_train_mfcc = np.array(y_train_mfcc)
```

ARCHITECTURE

In [9]:
```
def create_keras_model(model_input, num_classes):
    x = Conv2D(64, kernel_size=3, activation="relu",padding='same')(model_input)
    x = MaxPooling2D(pool_size=(2, 2))(x)
    x = Conv2D(128, kernel_size=3,padding='same', activation="relu")(x)
    x = MaxPooling2D(pool_size=(2, 2))(x)
    x = Conv2D(256, kernel_size=3,padding='same', activation="relu")(x)
    x = MaxPooling2D(pool_size=(2, 2))(x)
    x = Conv2D(256, kernel_size=3,padding='same', activation="relu")(x)
    x = MaxPooling2D(pool_size=(2, 2))(x)
    x = Flatten()(x)
    x = Dense(256, activation = 'relu')(x)
    x = Dropout(0.5)(x)
    x = Dense(num_classes, activation="softmax")(x)
    model = Model(model_input, x, name='model_1')
    return model
```

In [13]:

```
callbacks = [
    tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=10, verbose=1,
mode="min"),
    tf.keras.callbacks.ReduceLROnPlateau(monitor='val_loss',
                                mode='min',
                                verbose=1,
                                patience=5,
                                min_delta=0.0001,
                                factor=0.2
                               ),
]
```

MODEL SUMMARY:

```
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 20, 216, 1)]      0
_________________________________________________________________
conv2d (Conv2D)              (None, 20, 216, 64)       640
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 10, 108, 64)       0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 10, 108, 128)      73856
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 5, 54, 128)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 5, 54, 256)        295168
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 2, 27, 256)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 2, 27, 256)        590080
_________________________________________________________________
```

```
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 1, 13, 256)        0
_________________________________________________________________
flatten (Flatten)            (None, 3328)              0
_________________________________________________________________
dense (Dense)                (None, 256)               852224
_________________________________________________________________
dropout (Dropout)            (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 50)                12850
=================================================================
Total params: 1,824,818
Trainable params: 1,824,818
Non-trainable params: 0
_________________________________________________________________
```

COMPILING THE MODEL:

In [15]:
```
model_mfcc_1.compile(optimizer='adam',loss='sparse_categorical_crossentropy',metrics=['accuracy'])
```

In [16]:
```
# Training and Evaluation of the model
h1 = model_mfcc_1.fit(x_train_mfcc, y_train_mfcc,verbose=1, batch_size = 64 ,epochs=200,validation_split=0.1, callbacks = callbacks)
```

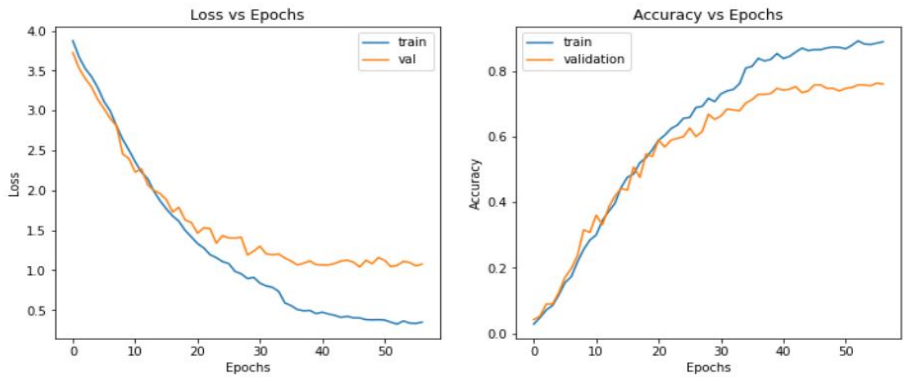
TEST RESULTS:

```
Epoch 53/200
54/54 [==============================] - 1s 16ms/step - loss: 0.3368 - accura
cy: 0.8897 - val_loss: 1.0606 - val_accuracy: 0.7579
Epoch 54/200
54/54 [==============================] - 1s 16ms/step - loss: 0.3579 - accura
cy: 0.8916 - val_loss: 1.1111 - val_accuracy: 0.7579
Epoch 55/200
54/54 [==============================] - 1s 16ms/step - loss: 0.3314 - accura
cy: 0.8799 - val_loss: 1.0971 - val_accuracy: 0.7553
Epoch 56/200
54/54 [==============================] - 1s 16ms/step - loss: 0.3328 - accura
cy: 0.8906 - val_loss: 1.0553 - val_accuracy: 0.7632
Epoch 57/200
54/54 [==============================] - 1s 16ms/step - loss: 0.3530 - accura
cy: 0.8843 - val_loss: 1.0778 - val_accuracy: 0.7605

Epoch 00057: ReduceLROnPlateau reducing learning rate to 8.000000525498762e-0
6.
Epoch 00057: early stopping
```

LOSS AND ACCURACY GRAPHS

```
In [18]: plot_metrics(h1)
```



INPUT: 3800X20X216 MFCC spectrogram images
OUTPUT: A model with learned weights

ARCHITECTURE:

```
    x = Conv2D(64, kernel_size=(3, 3), activation='relu',padding='same')(model_i
nput)
    x = BatchNormalization()(x)

    x = Conv2D(128, kernel_size=(3, 3), activation='relu',padding='same')(x)
    x = BatchNormalization()(x)

    x = Conv2D(128, kernel_size=(3, 3), activation='relu',padding='same')(x)
    x = BatchNormalization()(x)

    x = MaxPooling2D(pool_size=(4, 4))(x)
    x = Dropout(0.25)(x)

    x = Flatten()(x)

    x = Dense(256, activation='relu')(x)
    x = BatchNormalization()(x)
    x = Dropout(0.25)(x)

    x = Dense(256, activation='relu')(x)
    x = BatchNormalization()(x)
    x = Dropout(0.25)(x)

    x = Dense(64, activation='relu')(x)
    x = BatchNormalization()(x)
    x = Dropout(0.25)(x)

    x = Dense(num_classes, activation='softmax')(x)
    model = Model(model_input,x,name = 'model_2')
    return model
```

COMPILING THE MODEL

In [21]:
```
model_mfcc_2.compile(loss='sparse_categorical_crossentropy', optimizer=tf.keras.
optimizers.Adam(), metrics=['accuracy'])
```

In [22]:
```
h2 = model_mfcc_2.fit(x_train_mfcc, y_train_mfcc, batch_size=64, epochs=200, ver
bose=1, validation_split=0.1,callbacks = callbacks)
```

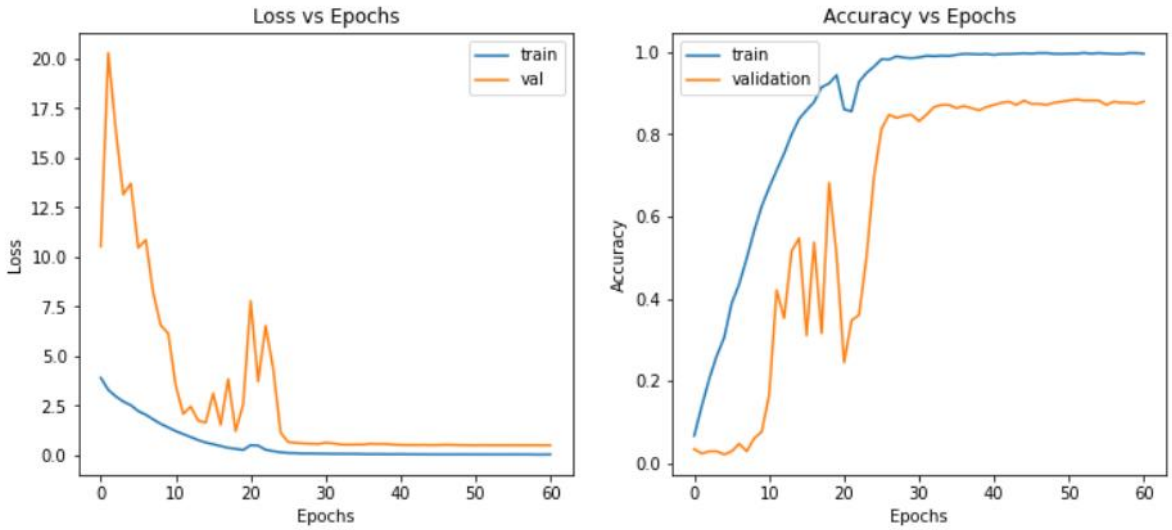
RESULTS

```
Epoch 60/200
54/54 [==============================] - 3s 48ms/step - loss: 0.0371 - accura
cy: 0.9972 - val_loss: 0.5062 - val_accuracy: 0.8737
Epoch 61/200
54/54 [==============================] - 3s 47ms/step - loss: 0.0411 - accura
cy: 0.9952 - val_loss: 0.5065 - val_accuracy: 0.8789

Epoch 00061: ReduceLROnPlateau reducing learning rate to 1.6000001778593287e-
06.
Epoch 00061: early stopping
```

LOSS & ACCURACY GRAPHS

```
]:
plot_metrics(h2)
```



INPUT: 3800X20X216 MFCC spectrogram images
OUTPUT: A model with learned weights