APS360 Project - Instrument Classifier

Disclaimer: While training the different models the accuracy and loss values would vary while re-training on the same parameters. Therefore, some of the numbers in this document may no longer match the values in the final report exactly because we re-compiled before submission.

In [ ]:
```
import os
import matplotlib.pyplot as plt
#Load librosa to convert our audio files
import librosa
import librosa.display
import pandas as pd
import numpy as np

import sklearn

from sklearn.metrics import accuracy_score

#to actually play the audio
import IPython.display as ipd

"""
path_audio_files = "/Test audio/"
audio_clips = os.listdir(path_audio_files)

#check number of files
print(len(audio_clips))
print(audio_clips)
"""
```

Out[ ]:
```
'\npath_audio_files = "/Test audio/"\naudio_clips = os.listdir(path_audio_fil
es)\n\n#check number of files\nprint(len(audio_clips))\nprint(audio_clips)\n'
```

Data Processing

old code for initial pre processing

In [ ]:
```
#df = pd.read_csv ('/content/train.csv')
#df
```

Get count of each instrument in csv

In [ ]:
```
#groupby_label = df.groupby(['label']).count()
#print(str(groupby_label.sort_values(by=['fname'], ascending= False)))
```

Choose instruments for classification

In [ ]:
```
#df = df[df['label'].isin(['Acoustic_guitar','Hi-hat','Double_bass','Saxophon
e','Clarinet', 'Cello','Trumpet','Bass_drum', 'Violin_or_fiddle', 'Flute'])]
```

make sure size and count matches

In [ ]:
```
#df.shape
```

get count of samples per instrument

In [ ]:
```
#chosen = df.groupby(['label']).count()
#chosen
```

Out[ ]:

label	fname	manually_verified
Acoustic_guitar	300	300
Bass_drum	300	300
Cello	300	300
Clarinet	300	300
Double_bass	300	300
Flute	300	300
Hi-hat	300	300
Saxophone	300	300
Trumpet	300	300
Violin_or_fiddle	300	300

Save csv file

In [ ]:
```
#df2.to_csv('/content/new2secs.csv',index=False)
```

upload new csv

In [ ]:
```
#df = pd.read_csv ('/content/new.csv')
#df
```

unzip audio files

In [ ]:
```
#df = pd.read_csv ('/content/new2secs.csv')
#df
```

Out[ ]:

	fname	label	manually_verified	Duration
0	00044347.wav	Hi-hat	0	14.00
1	001ca53d.wav	Saxophone	1	10.32
2	00353774.wav	Cello	1	4.52
3	003b91e8.wav	Cello	0	13.28
4	004ad66f.wav	Clarinet	0	7.00
...	...	...	...	...
2189	ff55a1e2.wav	Acoustic_guitar	0	14.66
2190	ff752a0c.wav	Clarinet	1	6.00
2191	ff875923.wav	Cello	0	11.84
2192	ff9c6c3f.wav	Trumpet	0	12.06
2193	ffc92b01.wav	Cello	1	6.24

2194 rows × 4 columns

In [ ]:
```
#!unzip /content/drive/MyDrive/aps360\ project/new.zip -d /content/audio/
```

Add Duration Attribute to CSV

In [ ]:
```
#path = path = "/content/audio/content/audio/audio_train/"
#df["Duration"] = ""


#for i in range(df.shape[0]):
#  df.at[i, "Duration"] = librosa.get_duration(filename= (path+df['fname'].ilo
c[i]))
```

In [ ]:
```
#df
```

Out[ ]:

	fname	label	manually_verified	Duration
0	00044347.wav	Hi-hat	0	14
1	001ca53d.wav	Saxophone	1	10.32
2	002d256b.wav	Trumpet	0	0.44
3	00353774.wav	Cello	1	4.52
4	003b91e8.wav	Cello	0	13.28
...	...	...	...	...
2995	ff752a0c.wav	Clarinet	1	6
2996	ff875923.wav	Cello	0	11.84
2997	ff9c6c3f.wav	Trumpet	0	12.06
2998	ffc92b01.wav	Cello	1	6.24
2999	fff37590.wav	Hi-hat	0	0.78

3000 rows × 4 columns

discard damples with duration less than 2 seconds

In [ ]:
```
#df = df[df['Duration'] >= 2.0]
```

get count after 2 second duration limit

In [ ]:
```
#df.groupby(['label']).count()
```

Out[ ]:

	fname	manually_verified	Duration
label			
Acoustic_guitar	275	275	275
Bass_drum	240	240	240
Cello	286	286	286
Clarinet	290	290	290
Double_bass	255	255	255
Flute	270	270	270
Hi-hat	282	282	282
Saxophone	250	250	250
Trumpet	257	257	257
Violin_or_fiddle	250	250	250

delete files that are not needed. only keep files that are in csv for the selected instruments. reduces time for donwload and upload, and zip and unzip

In [ ]:
```
#names = []

#for name in df['fname']:
 # names.append(name)

#print(len(names))
```

```
2194
```

In [ ]:
```
#path = "/content/audio/content/audio/audio_train/"

#remove=0
#for file in os.listdir(path):
#  if file not in names :
#    try:
#      remove+=1
#      os.remove(path+file)
#    except:
#      print("Error while deleting file : ", path)
```

ZIP new audio files

In [ ]:
```
#!zip -r /content/new2secs.zip /content/audio/content/audio/audio_train
```

In [ ]:
```
pip install scikit-learn
```

```
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.7/dist-
packages (0.22.2.post1)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-
packages (from scikit-learn) (1.0.1)
Requirement already satisfied: numpy>=1.11.0 in /usr/local/lib/python3.7/dist
-packages (from scikit-learn) (1.19.5)
Requirement already satisfied: scipy>=0.17.0 in /usr/local/lib/python3.7/dist
-packages (from scikit-learn) (1.4.1)
```

In [ ]:
```
df = pd.read_csv ('/content/new2secs.csv')
row, col = df.shape
```

get count of sampels per instrument

In [ ]:
```
chosen = df.groupby(['label']).count()
chosen
```

Out[ ]:

	fname	manually_verified	Duration
label			
Acoustic_guitar	275	275	275
Bass_drum	80	80	80
Cello	286	286	286
Clarinet	290	290	290
Double_bass	205	205	205
Flute	270	270	270
Hi-hat	141	141	141
Saxophone	250	250	250
Trumpet	207	207	207
Violin_or_fiddle	190	190	190

In [ ]:
```
temp = df
```

Duplicate Samples

In [ ]:
```
violin = 0
double_bass = 0
trumpet = 0

for i in df.values:
    if i[1] == 'Bass_drum':
      for x in range(2):
        new = pd.DataFrame([i], columns = ['fname','label','manually_verified'
,'Duration'])
        temp = temp.append(new,ignore_index=True)
    elif i[1] == 'Hi-hat':
      new = pd.DataFrame([i], columns = ['fname','label','manually_verified',
'Duration'])
      temp = temp.append(new,ignore_index=True)
    elif i[1] == 'Violin_or_fiddle' and violin <60 :
      new = pd.DataFrame([i], columns = ['fname','label','manually_verified',
'Duration'])
      temp = temp.append(new,ignore_index=True)
      violin += 1
    elif i[1] == 'Double_bass' and double_bass <50 :
      new = pd.DataFrame([i], columns = ['fname','label','manually_verified',
'Duration'])
      temp = temp.append(new,ignore_index=True)
      double_bass += 1
    elif i[1] == 'Trumpet' and trumpet <50 :
          new = pd.DataFrame([i], columns = ['fname','label','manually_verifie
d','Duration'])
          temp = temp.append(new,ignore_index=True)
          trumpet += 1
```

In [ ]:
```
df = temp
print(df.shape)
```

```
(2655, 4)
```

In [ ]:
```
chosen = df.groupby(['label']).count()
print(chosen)
```

```
                  fname  manually_verified  Duration
label
Acoustic_guitar     275                275       275
Bass_drum           240                240       240
Cello               286                286       286
Clarinet            290                290       290
Double_bass         255                255       255
Flute               270                270       270
Hi-hat              282                282       282
Saxophone           250                250       250
Trumpet             257                257       257
Violin_or_fiddle    250                250       250
```

Split data

70/15/15 split

```
In [ ]: train = round(0.7*row)
        validation = round(0.15*row)
        df = df.sample(frac=1)

        train_data = df[:train]
        validation_data = df[train:train+validation]
        test_data = df[row - validation:]
```

```
In [ ]: #print(train_data)
        #print(validation_data)
        #print(test_data)
        len(train_data)
        chosen = train_data.groupby(['label']).count()
        chosen
```

Mounting our Google Drive in order to unzip dataset with all audio clips of 2 seconds minimum length

```
In [ ]: from google.colab import drive
        drive.mount('/content/gdrive')
```

```
In [ ]: #!rm -rf /content/audio
```

```
In [ ]: #!rm -rf /content/test
        #!rm -rf /content/train
        #!rm -rf /content/valid
```

```
In [ ]: !unzip /content/gdrive/MyDrive/DATA/new2secs.zip -d /content/audio/
```

```
In [ ]: # Creating the separate files to make it easier to store our data for the tr
        aining, validation and testing set
        !mkdir train
        !mkdir valid
        !mkdir test
```

In [ ]:
```
# Defining the arrays that will store our audio wav files
train_names = []
valid_names = []
test_names = []

# Looping through each respective dataset and adding it to the corresponding a
rrays
for name in train_data['fname']:
  train_names.append(name)
for name in validation_data['fname']:
  valid_names.append(name)
for name in test_data['fname']:
  test_names.append(name)

# Test code to check if the sizes matched our dataset sizes
#print(len(train_names))
#print(len(valid_names))
#print(len(test_names))
```

The code below is to separate our training, validation and testing data in 3 separate folders to be able to organize our waveforms and heatmap images later in the code.

In [ ]:
```
# The code below is placing the wav files from the training, validation and te
sting data into their respective files on Colab
import glob
import shutil
path = "/content/audio/content/audio/content/audio/audio_train/*.*"
for file in glob.glob(path):
  if os.path.basename(file) in train_names:
    shutil.copy(file, "/content/train")

  if os.path.basename(file) in valid_names:
    shutil.copy(file, "/content/valid")

  if os.path.basename(file) in test_names:
    shutil.copy(file, "/content/test")
```

```
In [ ]: audio_train_files = "/content/train/"
        audio_valid_files = "/content/valid/"
        audio_test_files = "/content/test/"

        # Retrieving the name of each wav file in each set

        audio_train_clips = os.listdir(audio_train_files)
        audio_valid_clips = os.listdir(audio_valid_files)
        audio_test_clips = os.listdir(audio_test_files)



        # Testing the files for content and size
        #print(len(audio_train_clips))
        #print(len(audio_valid_clips))
        #print(len(audio_test_clips))
        #sum = len(audio_test_clips)+len(audio_train_clips)+len(audio_valid_clips)
        #print(sum)
```

```
In [ ]: #load audio files to visualize its waveform
        training_dataset= list()

        # Librosa.load used to retrieve the time-series values for each audio clip in
         the training data
        for i in range(len(audio_train_clips)):
          x, sr = librosa.load(audio_train_files+audio_train_clips[i])
          training_dataset.append(x)
```

```
In [ ]: #load audio files to visualize its waveform
        validation_dataset= list()

        # Librosa.load used to retrieve the time-series values for each audio clip in
         the validation data
        for i in range(len(audio_valid_clips)):
          x, sr = librosa.load(audio_valid_files+audio_valid_clips[i])
          validation_dataset.append(x)
```

```
In [ ]: #load audio files to visualize its waveform
        testing_dataset= list()

        # Librosa.load used to retrieve the time-series values for each audio clip in
         the testing data
        for i in range(len(audio_test_clips)):
          x, sr = librosa.load(audio_test_files+audio_test_clips[i])
          testing_dataset.append(x)
```

In [ ]:
```
# Converting our arrays to numpy arrays
traing_dataset=np.array(training_dataset)
validation_dataset=np.array(validation_dataset)
testing_dataset=np.array(validation_dataset)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: VisibleDeprec
ationWarning: Creating an ndarray from ragged nested sequences (which is a li
st-or-tuple of lists-or-tuples-or ndarrays with different lengths or shapes)
is deprecated. If you meant to do this, you must specify 'dtype=object' when
creating the ndarray

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: VisibleDeprec
ationWarning: Creating an ndarray from ragged nested sequences (which is a li
st-or-tuple of lists-or-tuples-or ndarrays with different lengths or shapes)
is deprecated. If you meant to do this, you must specify 'dtype=object' when
creating the ndarray
  This is separate from the ipykernel package so we can avoid doing imports u
ntil
```

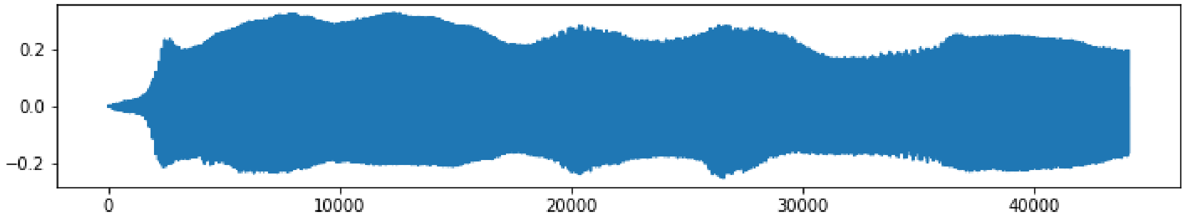
The code below is used to remove all of the time-series values after 2 seconds for every audio clip in each dataset. This is to ensure the inputs to our CNN will all be of the same size.

In [ ]:
```
training_audio_waves= list()
valid_audio_waves= list()
test_audio_waves= list()

# Remove values after 2 seconds. The 44100 is equivalent to the sampling rate
 x 2 seconds. (our sampling rate was 22050)
for i in range(len(training_dataset)):
  training_audio_waves.append(training_dataset[i][:44100])

for i in range(len(validation_dataset)):
  valid_audio_waves.append(validation_dataset[i][:44100])

for i in range(len(testing_dataset)):
  test_audio_waves.append(testing_dataset[i][:44100])
```

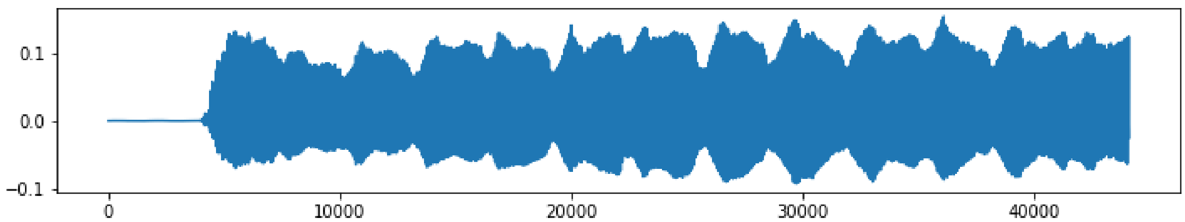
In [ ]:
```
# Convert our final data sets to numpy arrays
training_audio_waves= np.array(training_audio_waves)
valid_audio_waves= np.array(valid_audio_waves)
test_audio_waves= np.array(test_audio_waves)
```

Below are plots of the waveforms for random audio clips in each set.

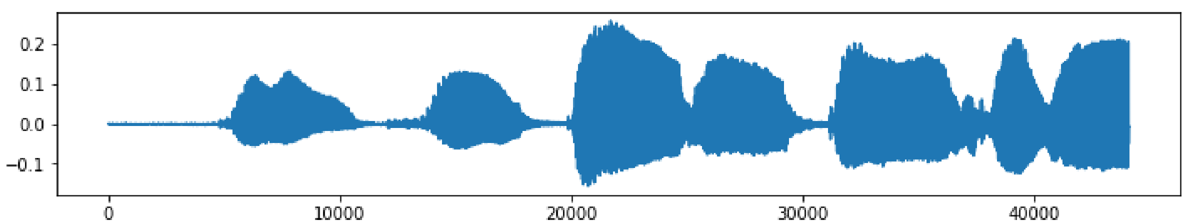
In [ ]:
```
# Display the waveform for the 22nd audio clip in the training dataset
plt.figure(figsize=(12,2))
plt.plot(training_audio_waves[22])
plt.show()
```



In [ ]:
```
# Display the waveform for the 10th audio clip in the validation dataset
plt.figure(figsize=(12,2))
plt.plot(valid_audio_waves[10])
plt.show()
```



In [ ]:
```
# Display the waveform for the 30th audio clip in the testing set
plt.figure(figsize=(12,2))
plt.plot(test_audio_waves[30])
plt.show()
```



The code below is to create the arrays for the labels in each dataset to be used later in creating our one-hot encodings.

In [ ]:
```
labels_train_array= list()
for i in range(len(audio_train_clips)):
  labels_train_array.append(df[df['fname']== audio_train_clips[i]]['label'].va
lues[0])

labels_train_array = np.array(labels_train_array)
print(len(labels_train_array))
print(labels_train_array)
```

In [ ]:
```
labels_valid_array= list()
for i in range(len(audio_valid_clips)):
  labels_valid_array.append(df[df['fname']== audio_valid_clips[i]]['label'].values[0])

labels_valid_array = np.array(labels_valid_array)

print(len(labels_valid_array))
print(labels_valid_array)
```

In [ ]:
```
labels_test_array= list()
for i in range(len(audio_test_clips)):
  labels_test_array.append(df[df['fname']== audio_test_clips[i]]['label'].values[0])

labels_test_array = np.array(labels_test_array)
print(len(labels_test_array))
print(labels_train_array)
```

The code below is where we take the waveforms of each audio clip in all of our sets and apply the mfcc function to extract features and create the final heatmaps.

In [ ]:
```
pip install SpeechRecognition
```

```
Collecting SpeechRecognition
  Downloading https://files.pythonhosted.org/packages/26/e1/7f5678cd94ec1234269d23756dbdaa4c8cfaed973412f88ae8adf7893a50/SpeechRecognition-3.8.1-py2.py3-none-any.whl (32.8MB)
     |████████████████████████████████| 32.8MB 118kB/s
Installing collected packages: SpeechRecognition
Successfully installed SpeechRecognition-3.8.1
```

In [ ]:
```
pip install python_speech_features
```

```
Collecting python_speech_features
  Downloading https://files.pythonhosted.org/packages/ff/d1/94c59e20a2631985fbd2124c45177abaa9e0a4eee8ba8a305aa26fc02a8e/python_speech_features-0.6.tar.gz
Building wheels for collected packages: python-speech-features
  Building wheel for python-speech-features (setup.py) ... done
  Created wheel for python-speech-features: filename=python_speech_features-0.6-cp37-none-any.whl size=5887 sha256=b5632a10b65e5d03e703e4f200afaeb9c0ef17a5cc45251fbdd1d490e8686854
  Stored in directory: /root/.cache/pip/wheels/3c/42/7c/f60e9d1b40015cd69b213ad90f7c18a9264cd745b9888134be
Successfully built python-speech-features
Installing collected packages: python-speech-features
Successfully installed python-speech-features-0.6
```

In [ ]:
```
import os
import librosa
import pickle
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from tqdm import tqdm
from python_speech_features import mfcc
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from keras.layers import Conv2D, MaxPool2D, Flatten, Dense, Dropout
from keras.models import Sequential
```

In [ ]:
```
# Defining the array to store our values for the MFCC representations of the t
raining audio clips to create our heat maps
mfcc_features_training_dataset = list()

for i in range(len(training_audio_waves)):
  mfcc_features_training_dataset.append(mfcc(training_audio_waves[i]))


# Test code to check if sizes match
#print(training_audio_waves.shape)
#print(mfcc_features_training_dataset.shape)
```

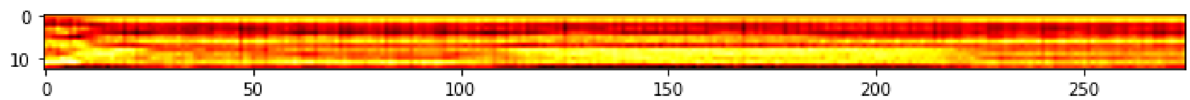
In [ ]:
```
# Defining the array to store our values for the MFCC representations of the v
alidation audio clips to create our heat maps
mfcc_features_validation_dataset = list()


for i in range(len(valid_audio_waves)):
  mfcc_features_validation_dataset.append(mfcc(valid_audio_waves[i]))

# Test code to check if sizes match
#print(valid_audio_waves.shape)
#print(mfcc_features_validation_dataset.shape)
```

In [ ]:
```
# Defining the array to store our values for the MFCC representations of the t
esting audio clips to create our heat maps
mfcc_features_testing_dataset = list()

for i in range(len(test_audio_waves)):
  mfcc_features_testing_dataset.append(mfcc(test_audio_waves[0]))

# Test code to check if sizes match
#print(test_audio_waves.shape)
#print(mfcc_features_testing_dataset.shape)
```

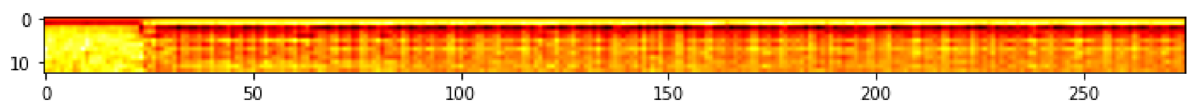
In [ ]:
```
# Convert heatmap arrays to numpy arrays
mfcc_features_validation_dataset= np.array(mfcc_features_validation_dataset)
mfcc_features_training_dataset= np.array(mfcc_features_training_dataset)
mfcc_features_testing_dataset= np.array(mfcc_features_testing_dataset)

# Test code
#print(mfcc_features_a.shape)
#print(mfcc_features_b.shape)
#print(mfcc_features_c.shape)
```

Below are the plots for our final heatmap (images) that we will use as inputs for our CNN.

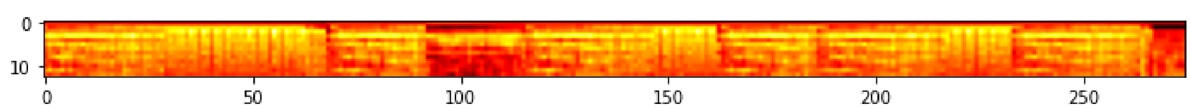
In [ ]:
```
# Display the mfcc features of the 22nd audio clip in the training dataset as
 a heatmap
plt.figure(figsize=(12, 2))
plt.imshow(mfcc_features_training_dataset[22].T, cmap='hot')
plt.show()
```



In [ ]:
```
# Display the mfcc features of the 10th audio clip in the validation dataset a
s a heatmap
plt.figure(figsize=(12, 2))
plt.imshow(mfcc_features_validation_dataset[10].T, cmap='hot')
plt.show()
```



In [ ]:
```
# Display the mfcc features of the 30th audio clip in the testing dataset as a
heatmap
plt.figure(figsize=(12, 2))
plt.imshow(mfcc_features_training_dataset[30].T, cmap='hot')
plt.show()
```



Below we define our hot encodings for the training, validation and testing sets.

In [ ]:
```
label_encoder_a = LabelEncoder()
label_encoded_a = label_encoder_a.fit_transform(labels_train_array)
#print(label_encoded_a)

label_encoded_a = label_encoded_a[:, np.newaxis]
label_encoded_a

one_hot_encoder_a = OneHotEncoder(sparse=False)
one_hot_encoded_a = one_hot_encoder_a.fit_transform(label_encoded_a)
one_hot_encoded_a
print(one_hot_encoded_a.shape)
```

```
(1385, 10)
```

In [ ]:
```
label_encoder_b = LabelEncoder()
label_encoded_b = label_encoder_b.fit_transform(labels_valid_array)
#print(label_encoded_b)

label_encoded_b = label_encoded_b[:, np.newaxis]
label_encoded_b

one_hot_encoder_b = OneHotEncoder(sparse=False)
one_hot_encoded_b = one_hot_encoder_b.fit_transform(label_encoded_b)
one_hot_encoded_b
print(one_hot_encoded_b.shape)
```

```
(323, 10)
```

In [ ]:
```
label_encoder_c = LabelEncoder()
label_encoded_c = label_encoder_c.fit_transform(labels_test_array)
#print(label_encoded_c)

label_encoded_c = label_encoded_c[:, np.newaxis]
label_encoded_c

one_hot_encoder_c = OneHotEncoder(sparse=False)
one_hot_encoded_c = one_hot_encoder_c.fit_transform(label_encoded_c)
one_hot_encoded_c
print(one_hot_encoded_c.shape)
```

```
(752, 10)
```

Baseline Model

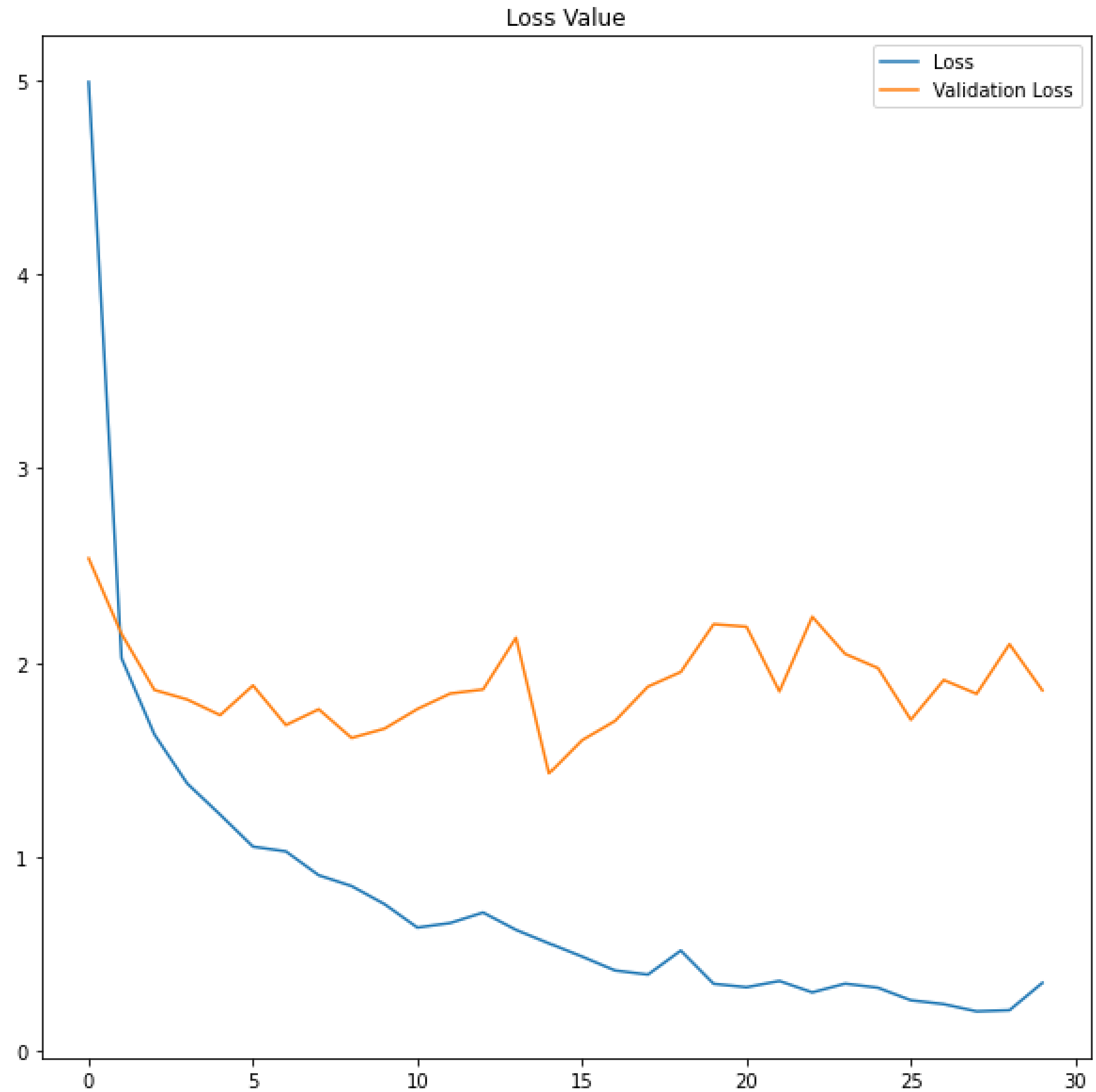
Baseline Model: Fully Connected Layer

In [97]:
```
# Train/Validation/Test split
X_train, X_validation, y_train, Y_validation = mfcc_features_training_dataset,
mfcc_features_validation_dataset, one_hot_encoded_a, one_hot_encoded_b
X_test, y_test = mfcc_features_testing_dataset, one_hot_encoded_c

# Defining input shape for the neural network
input_shape = (X_train.shape[1], X_train.shape[2], 1)

# Reshape X_train and X_validation such that they are having the same shape as
the input shape
X_train = X_train.reshape(X_train.shape[0], X_train.shape[1], X_train.shape[2
], 1)
X_validation = X_validation.reshape(X_validation.shape[0], X_validation.shape[
1], X_validation.shape[2], 1)
X_test = X_test.reshape(X_test.shape[0], X_test.shape[1], X_test.shape[2], 1)

# Constructing the neural network architecture
model = Sequential()
model.add(Flatten())
model.add(Dense(128, activation = 'relu'))
model.add(Dense(64, activation = 'relu'))
model.add(Dense(32, activation = 'relu'))
model.add(Dense(10, activation = 'softmax'))

model.compile(loss = 'categorical_crossentropy',
     optimizer = 'adam',
     metrics = ['acc'])

# Training the model
history = model.fit(X_train, y_train, epochs = 30, validation_data = (X_valida
tion, Y_validation))

# Displaying loss values
plt.figure(figsize = (10, 10))
plt.title('Loss Value')
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.legend(['Loss', 'Validation Loss'])
print('Loss:', history.history['loss'][-1])
print('Validation Loss:', history.history['val_loss'][-1])
plt.show()

# Displaying accuracy scores
plt.figure(figsize=(10, 10))
plt.title('Accuracy')
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.legend(['Accuracy', 'Validation Accuracy'])
print('Accuracy:', history.history['acc'][-1])
print('Validation Accuracy:', history.history['val_acc'][-1])
plt.show()

# Model evaluation
predictions = model.predict(X_validation)

predictions = np.argmax(predictions, axis=1)
```

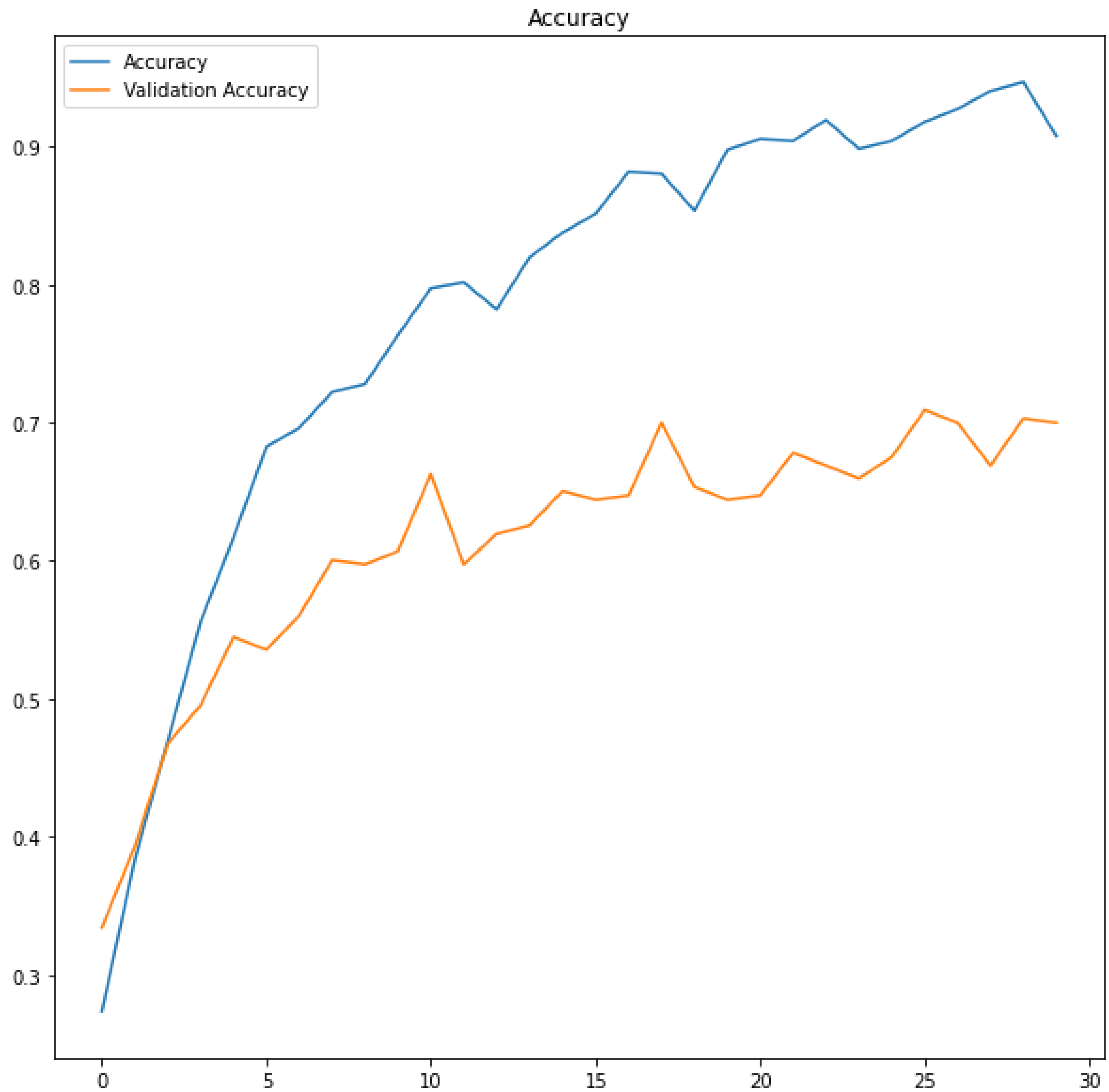
```python
Y_validation = one_hot_encoder_b.inverse_transform(Y_validation)

# Creating confusion matrix
cm = confusion_matrix(Y_validation, predictions)
plt.figure(figsize = (10, 10))
sns.heatmap(cm, annot = True, xticklabels = label_encoder_a.classes_, yticklab
els = label_encoder_b.classes_, fmt = 'd', cmap = plt.cm.Blues, cbar = False)
plt.xlabel('Predicted Label')
plt.ylabel('Actual Label')
plt.show()
```

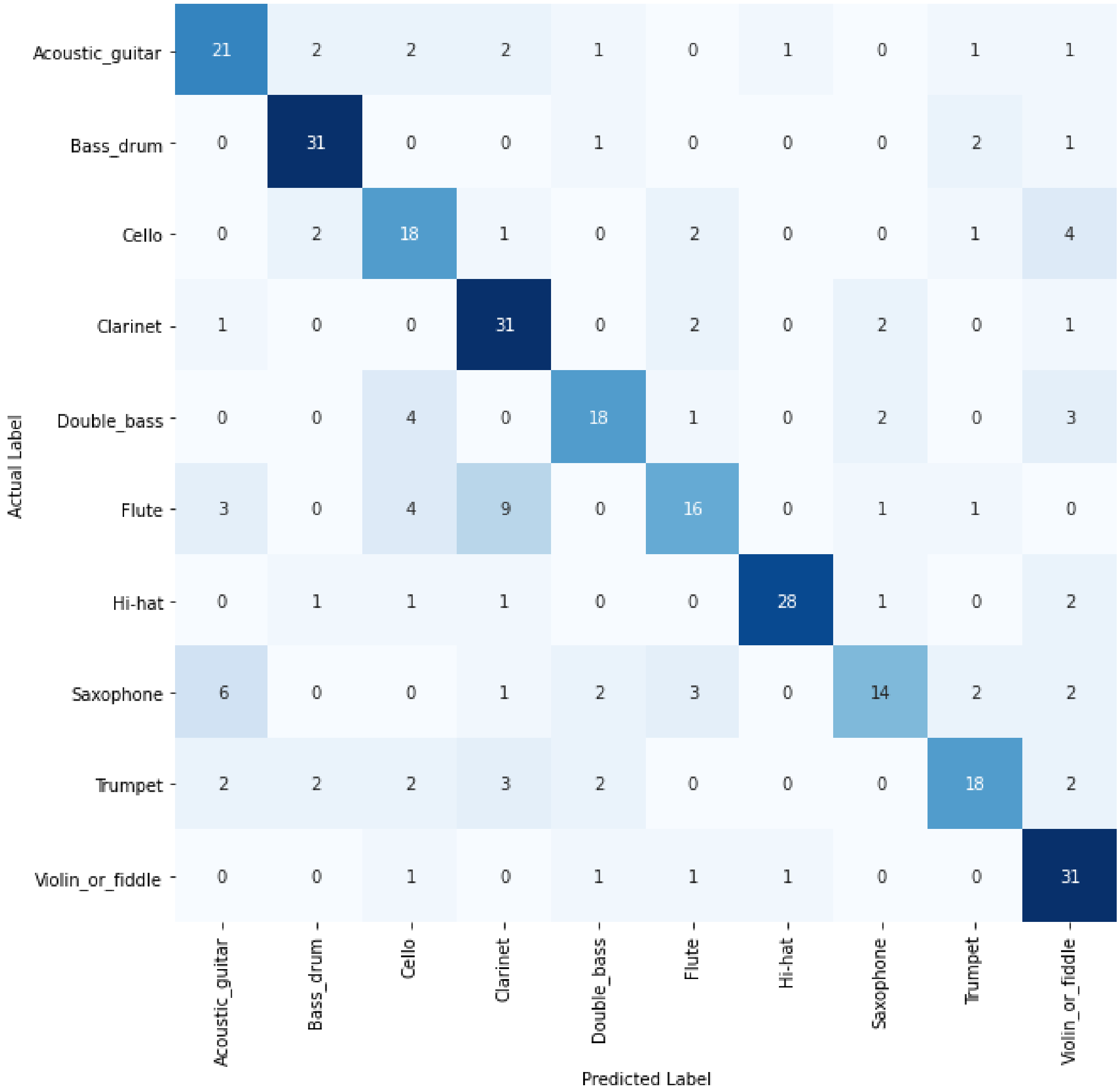
```
Epoch 1/30
44/44 [==============================] - 1s 11ms/step - loss: 6.9262 - acc:
0.2535 - val_loss: 2.5363 - val_acc: 0.3344
Epoch 2/30
44/44 [==============================] - 0s 7ms/step - loss: 2.1870 - acc: 0.
3357 - val_loss: 2.1475 - val_acc: 0.3932
Epoch 3/30
44/44 [==============================] - 0s 7ms/step - loss: 1.6346 - acc: 0.
4690 - val_loss: 1.8590 - val_acc: 0.4675
Epoch 4/30
44/44 [==============================] - 0s 7ms/step - loss: 1.3916 - acc: 0.
5465 - val_loss: 1.8095 - val_acc: 0.4954
Epoch 5/30
44/44 [==============================] - 0s 7ms/step - loss: 1.2374 - acc: 0.
6042 - val_loss: 1.7292 - val_acc: 0.5449
Epoch 6/30
44/44 [==============================] - 0s 7ms/step - loss: 1.0646 - acc: 0.
6748 - val_loss: 1.8819 - val_acc: 0.5356
Epoch 7/30
44/44 [==============================] - 0s 7ms/step - loss: 0.8919 - acc: 0.
7229 - val_loss: 1.6780 - val_acc: 0.5604
Epoch 8/30
44/44 [==============================] - 0s 8ms/step - loss: 0.9390 - acc: 0.
7029 - val_loss: 1.7594 - val_acc: 0.6006
Epoch 9/30
44/44 [==============================] - 0s 7ms/step - loss: 0.8296 - acc: 0.
7354 - val_loss: 1.6117 - val_acc: 0.5975
Epoch 10/30
44/44 [==============================] - 0s 7ms/step - loss: 0.7724 - acc: 0.
7478 - val_loss: 1.6596 - val_acc: 0.6068
Epoch 11/30
44/44 [==============================] - 0s 7ms/step - loss: 0.5957 - acc: 0.
8007 - val_loss: 1.7607 - val_acc: 0.6625
Epoch 12/30
44/44 [==============================] - 0s 7ms/step - loss: 0.6084 - acc: 0.
8069 - val_loss: 1.8404 - val_acc: 0.5975
Epoch 13/30
44/44 [==============================] - 0s 7ms/step - loss: 0.6698 - acc: 0.
7966 - val_loss: 1.8614 - val_acc: 0.6192
Epoch 14/30
44/44 [==============================] - 0s 7ms/step - loss: 0.5949 - acc: 0.
8279 - val_loss: 2.1277 - val_acc: 0.6254
Epoch 15/30
44/44 [==============================] - 0s 7ms/step - loss: 0.5675 - acc: 0.
8377 - val_loss: 1.4296 - val_acc: 0.6502
Epoch 16/30
44/44 [==============================] - 0s 7ms/step - loss: 0.4589 - acc: 0.
8615 - val_loss: 1.6003 - val_acc: 0.6440
Epoch 17/30
44/44 [==============================] - 0s 7ms/step - loss: 0.3602 - acc: 0.
8865 - val_loss: 1.6994 - val_acc: 0.6471
Epoch 18/30
44/44 [==============================] - 0s 7ms/step - loss: 0.3598 - acc: 0.
8857 - val_loss: 1.8759 - val_acc: 0.6997
Epoch 19/30
44/44 [==============================] - 0s 7ms/step - loss: 0.4823 - acc: 0.
8638 - val_loss: 1.9513 - val_acc: 0.6533
```

```
Epoch 20/30
44/44 [==============================] - 0s 7ms/step - loss: 0.3324 - acc: 0.
9101 - val_loss: 2.1977 - val_acc: 0.6440
Epoch 21/30
44/44 [==============================] - 0s 7ms/step - loss: 0.3450 - acc: 0.
8965 - val_loss: 2.1843 - val_acc: 0.6471
Epoch 22/30
44/44 [==============================] - 0s 7ms/step - loss: 0.3309 - acc: 0.
9146 - val_loss: 1.8513 - val_acc: 0.6780
Epoch 23/30
44/44 [==============================] - 0s 7ms/step - loss: 0.2878 - acc: 0.
9149 - val_loss: 2.2358 - val_acc: 0.6687
Epoch 24/30
44/44 [==============================] - 0s 7ms/step - loss: 0.3450 - acc: 0.
8977 - val_loss: 2.0444 - val_acc: 0.6594
Epoch 25/30
44/44 [==============================] - 0s 7ms/step - loss: 0.3319 - acc: 0.
9036 - val_loss: 1.9713 - val_acc: 0.6749
Epoch 26/30
44/44 [==============================] - 0s 7ms/step - loss: 0.2563 - acc: 0.
9201 - val_loss: 1.7052 - val_acc: 0.7090
Epoch 27/30
44/44 [==============================] - 0s 7ms/step - loss: 0.2786 - acc: 0.
9221 - val_loss: 1.9100 - val_acc: 0.6997
Epoch 28/30
44/44 [==============================] - 0s 7ms/step - loss: 0.1909 - acc: 0.
9399 - val_loss: 1.8390 - val_acc: 0.6687
Epoch 29/30
44/44 [==============================] - 0s 7ms/step - loss: 0.2123 - acc: 0.
9447 - val_loss: 2.0946 - val_acc: 0.7028
Epoch 30/30
44/44 [==============================] - 0s 7ms/step - loss: 0.3130 - acc: 0.
9177 - val_loss: 1.8569 - val_acc: 0.6997
Loss: 0.35027649998664856
Validation Loss: 1.8569003343582153
```




```
Accuracy: 0.9075812101364136
Validation Accuracy: 0.6996904015541077
```


Accuracy
Accuracy
Validation Accuracy
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0
5
10
15
20
25
30



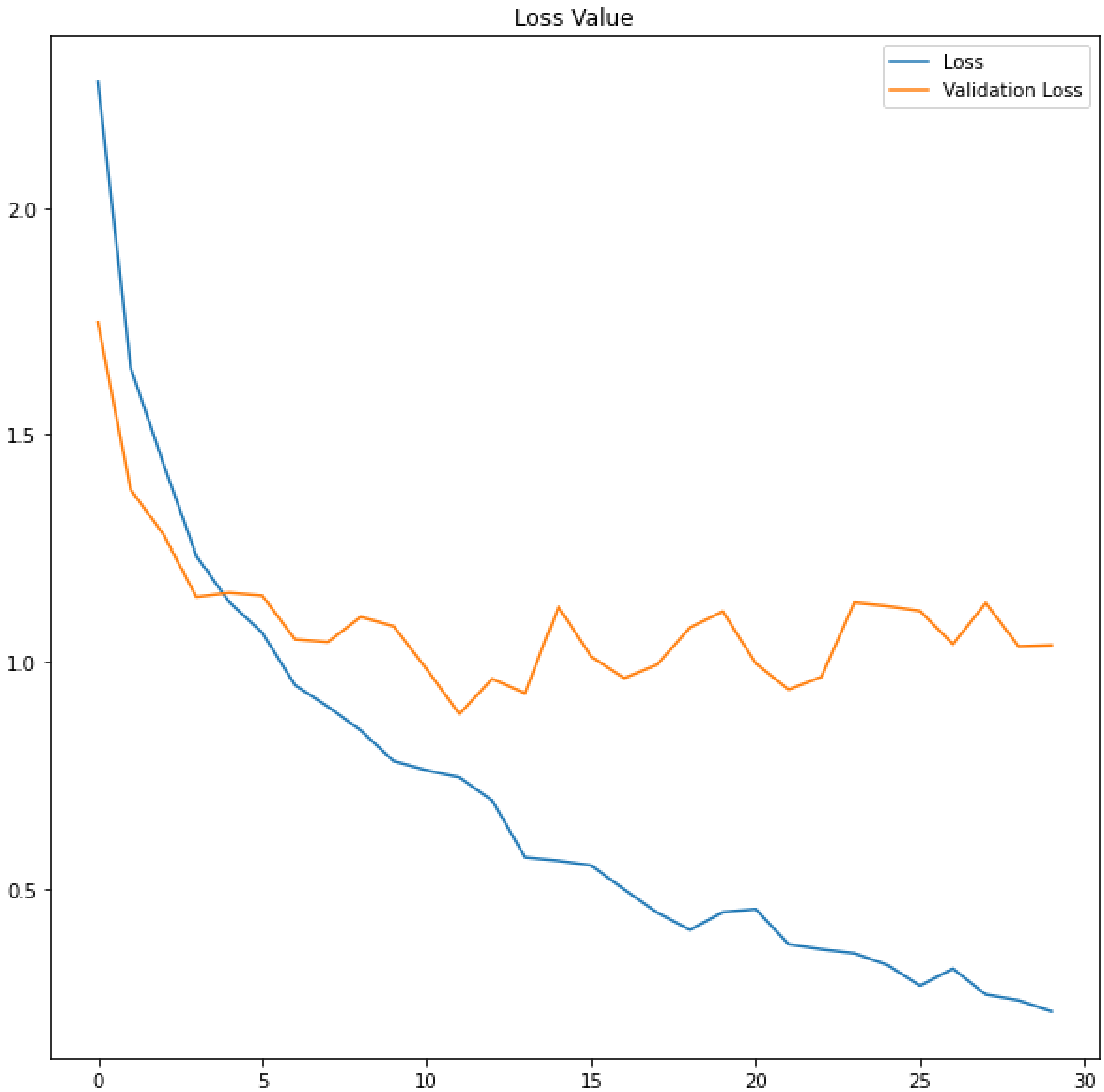
Final Primary Model

In [99]:
```
# Train/Validation/Test split
X_train, X_validation, y_train, Y_validation = mfcc_features_training_dataset,
mfcc_features_validation_dataset, one_hot_encoded_a, one_hot_encoded_b
X_test, y_test = mfcc_features_testing_dataset, one_hot_encoded_c

# Defining input shape for the neural network
input_shape = (X_train.shape[1], X_train.shape[2], 1)

# Reshape X_train and X_validation such that they are having the same shape as
the input shape
X_train = X_train.reshape(X_train.shape[0], X_train.shape[1], X_train.shape[2
], 1)
X_validation = X_validation.reshape(X_validation.shape[0], X_validation.shape[
1], X_validation.shape[2], 1)
X_test = X_test.reshape(X_test.shape[0], X_test.shape[1], X_test.shape[2], 1)

# Constructing the neural network architecture
model = Sequential()
model.add(Conv2D(32, (4, 4), activation ='relu', strides=(2, 2),
    padding='same', input_shape=input_shape))
model.add(MaxPool2D((2, 2)))
model.add(Conv2D(64, (4, 4), activation='relu', strides=(2, 2),
    padding='same'))
model.add(MaxPool2D((2, 2)))
model.add(Dropout(0.5))
model.add(Flatten())
model.add(Dense(128, activation = 'relu'))
model.add(Dense(64, activation = 'relu'))
model.add(Dense(10, activation = 'softmax'))

model.compile(loss='categorical_crossentropy',
     optimizer='adam',
     metrics=['acc'])

# Training the model
history = model.fit(X_train, y_train, epochs=30, validation_data=(X_validation
, Y_validation))

# Displaying loss values
plt.figure(figsize=(10, 10))
plt.title('Loss Value')
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.legend(['Loss', 'Validation Loss'])
print('Loss:', history.history['loss'][-1])
print('Validation Loss:', history.history['val_loss'][-1])
plt.show()

# Displaying accuracy scores
plt.figure(figsize=(10, 10))
plt.title('Accuracy')
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.legend(['Accuracy', 'Validation Accuracy'])
print('Accuracy:', history.history['acc'][-1])
```

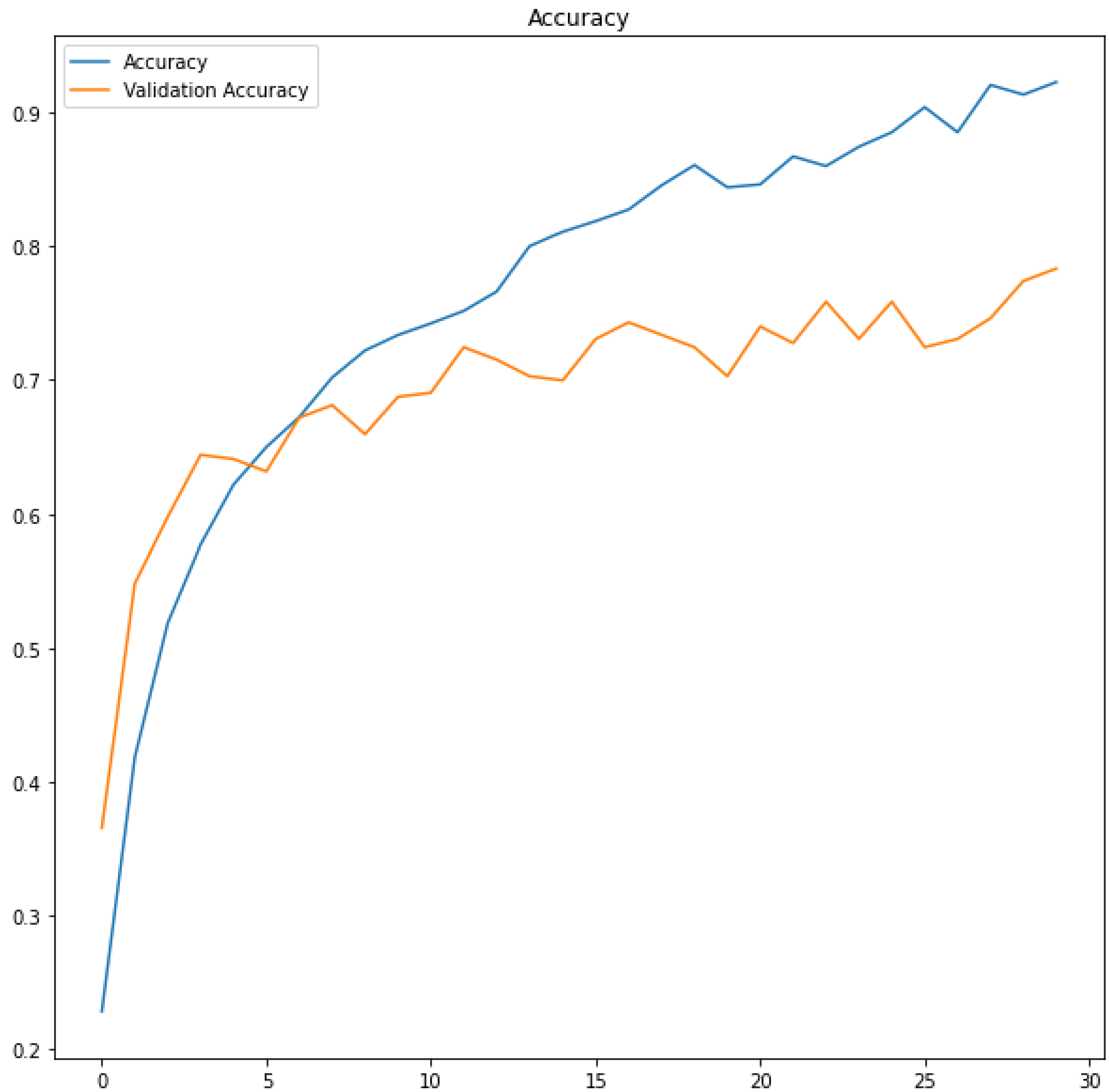
```
print('Validation Accuracy:', history.history['val_acc'][-1])
plt.show()
```

```
Epoch 1/30
44/44 [==============================] - 3s 44ms/step - loss: 2.7414 - acc:
0.1625 - val_loss: 1.7473 - val_acc: 0.3653
Epoch 2/30
44/44 [==============================] - 2s 39ms/step - loss: 1.7235 - acc:
0.3824 - val_loss: 1.3784 - val_acc: 0.5480
Epoch 3/30
44/44 [==============================] - 2s 39ms/step - loss: 1.4628 - acc:
0.5027 - val_loss: 1.2798 - val_acc: 0.5975
Epoch 4/30
44/44 [==============================] - 2s 39ms/step - loss: 1.2489 - acc:
0.5672 - val_loss: 1.1436 - val_acc: 0.6440
Epoch 5/30
44/44 [==============================] - 2s 39ms/step - loss: 1.1129 - acc:
0.6252 - val_loss: 1.1529 - val_acc: 0.6409
Epoch 6/30
44/44 [==============================] - 2s 39ms/step - loss: 0.9807 - acc:
0.6731 - val_loss: 1.1464 - val_acc: 0.6316
Epoch 7/30
44/44 [==============================] - 2s 40ms/step - loss: 0.9518 - acc:
0.6691 - val_loss: 1.0499 - val_acc: 0.6718
Epoch 8/30
44/44 [==============================] - 2s 40ms/step - loss: 0.9007 - acc:
0.7007 - val_loss: 1.0440 - val_acc: 0.6811
Epoch 9/30
44/44 [==============================] - 2s 40ms/step - loss: 0.8423 - acc:
0.7267 - val_loss: 1.0994 - val_acc: 0.6594
Epoch 10/30
44/44 [==============================] - 2s 40ms/step - loss: 0.7461 - acc:
0.7389 - val_loss: 1.0787 - val_acc: 0.6873
Epoch 11/30
44/44 [==============================] - 2s 40ms/step - loss: 0.7420 - acc:
0.7455 - val_loss: 0.9846 - val_acc: 0.6904
Epoch 12/30
44/44 [==============================] - 2s 39ms/step - loss: 0.6941 - acc:
0.7717 - val_loss: 0.8854 - val_acc: 0.7245
Epoch 13/30
44/44 [==============================] - 2s 40ms/step - loss: 0.7121 - acc:
0.7730 - val_loss: 0.9629 - val_acc: 0.7152
Epoch 14/30
44/44 [==============================] - 2s 39ms/step - loss: 0.5467 - acc:
0.8054 - val_loss: 0.9312 - val_acc: 0.7028
Epoch 15/30
44/44 [==============================] - 2s 40ms/step - loss: 0.5687 - acc:
0.8080 - val_loss: 1.1211 - val_acc: 0.6997
Epoch 16/30
44/44 [==============================] - 2s 40ms/step - loss: 0.5556 - acc:
0.8175 - val_loss: 1.0116 - val_acc: 0.7307
Epoch 17/30
44/44 [==============================] - 2s 39ms/step - loss: 0.5167 - acc:
0.8248 - val_loss: 0.9644 - val_acc: 0.7430
Epoch 18/30
44/44 [==============================] - 2s 41ms/step - loss: 0.4211 - acc:
0.8684 - val_loss: 0.9940 - val_acc: 0.7337
Epoch 19/30
44/44 [==============================] - 2s 39ms/step - loss: 0.4079 - acc:
0.8569 - val_loss: 1.0759 - val_acc: 0.7245
```

```
Epoch 20/30
44/44 [==============================] - 2s 39ms/step - loss: 0.4297 - acc: 0.8534 - val_loss: 1.1113 - val_acc: 0.7028
Epoch 21/30
44/44 [==============================] - 2s 39ms/step - loss: 0.4796 - acc: 0.8439 - val_loss: 0.9971 - val_acc: 0.7399
Epoch 22/30
44/44 [==============================] - 2s 38ms/step - loss: 0.3502 - acc: 0.8724 - val_loss: 0.9394 - val_acc: 0.7276
Epoch 23/30
44/44 [==============================] - 2s 39ms/step - loss: 0.3996 - acc: 0.8555 - val_loss: 0.9672 - val_acc: 0.7585
Epoch 24/30
44/44 [==============================] - 2s 41ms/step - loss: 0.3627 - acc: 0.8712 - val_loss: 1.1309 - val_acc: 0.7307
Epoch 25/30
44/44 [==============================] - 2s 39ms/step - loss: 0.3556 - acc: 0.8803 - val_loss: 1.1230 - val_acc: 0.7585
Epoch 26/30
44/44 [==============================] - 2s 39ms/step - loss: 0.2914 - acc: 0.9036 - val_loss: 1.1123 - val_acc: 0.7245
Epoch 27/30
44/44 [==============================] - 2s 39ms/step - loss: 0.3417 - acc: 0.8770 - val_loss: 1.0393 - val_acc: 0.7307
Epoch 28/30
44/44 [==============================] - 2s 39ms/step - loss: 0.2698 - acc: 0.9204 - val_loss: 1.1300 - val_acc: 0.7461
Epoch 29/30
44/44 [==============================] - 2s 39ms/step - loss: 0.2662 - acc: 0.9064 - val_loss: 1.0341 - val_acc: 0.7740
Epoch 30/30
44/44 [==============================] - 2s 40ms/step - loss: 0.2186 - acc: 0.9291 - val_loss: 1.0369 - val_acc: 0.7833
Loss: 0.23020996153354645
Validation Loss: 1.0368684530258179
```



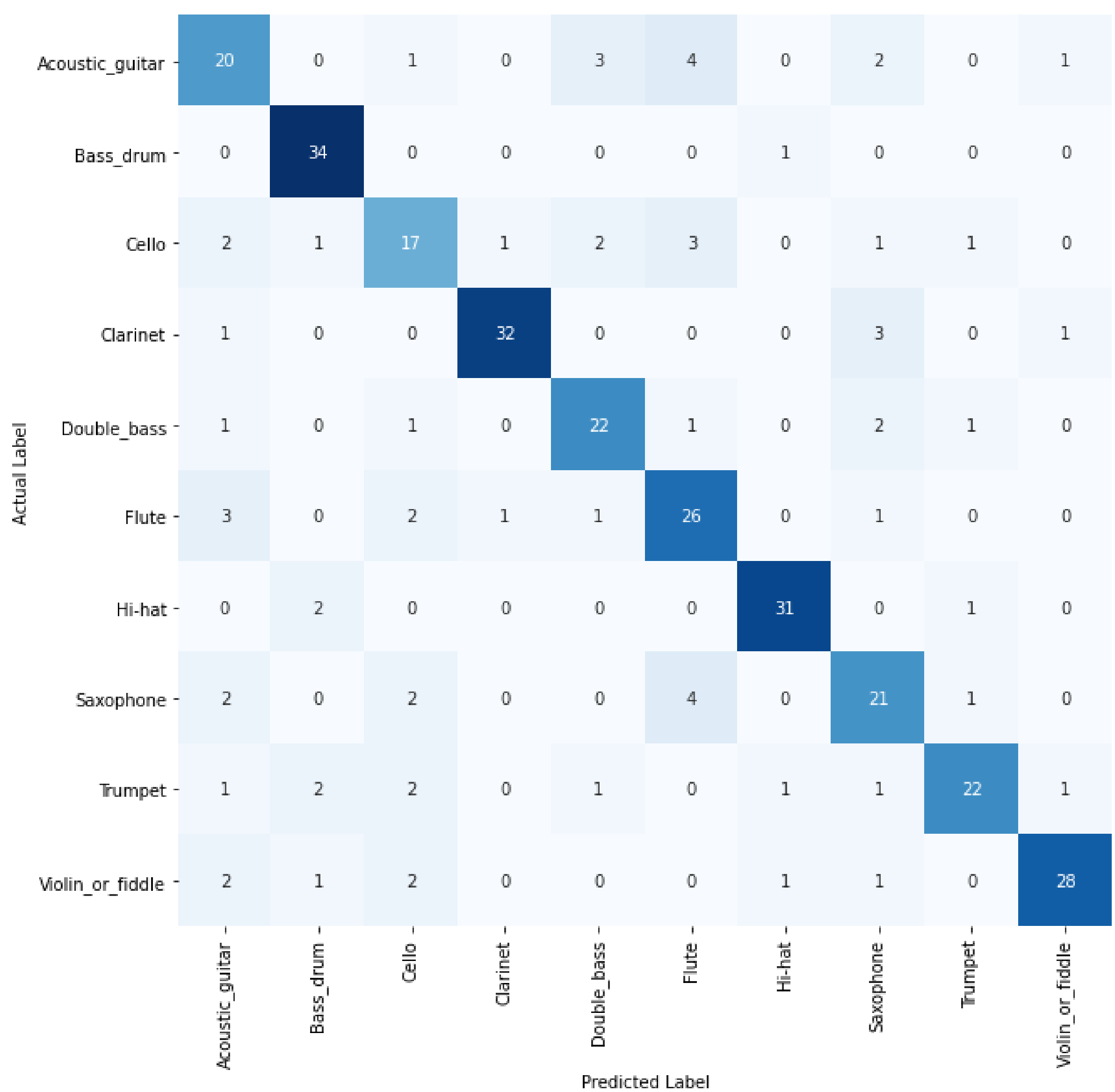

```
Accuracy: 0.92274367809295 65
Validation Accuracy: 0.7832817435264587
```



Final Accuracy

In [101]:
```
# Model evaluation on unseen data (testing data)
predictions = model.predict(X_test)

predictions = np.argmax(predictions, axis=1)
y_test = one_hot_encoder_b.inverse_transform(y_test)

ans = sklearn.metrics.accuracy_score(y_test, predictions, normalize=True, samp
le_weight=None)
print("Testing accuracy = ", ans)

# Creating confusion matrix
cm = confusion_matrix(y_test, predictions)
plt.figure(figsize=(10,10))
sns.heatmap(cm, annot=True, xticklabels=label_encoder_a.classes_, yticklabels=
label_encoder_b.classes_, fmt='d', cmap=plt.cm.Blues, cbar=False)
plt.xlabel('Predicted Label')
plt.ylabel('Actual Label')
plt.show()
```

```
Testing accuracy =  0.7832817337461301
```



Tuning Process

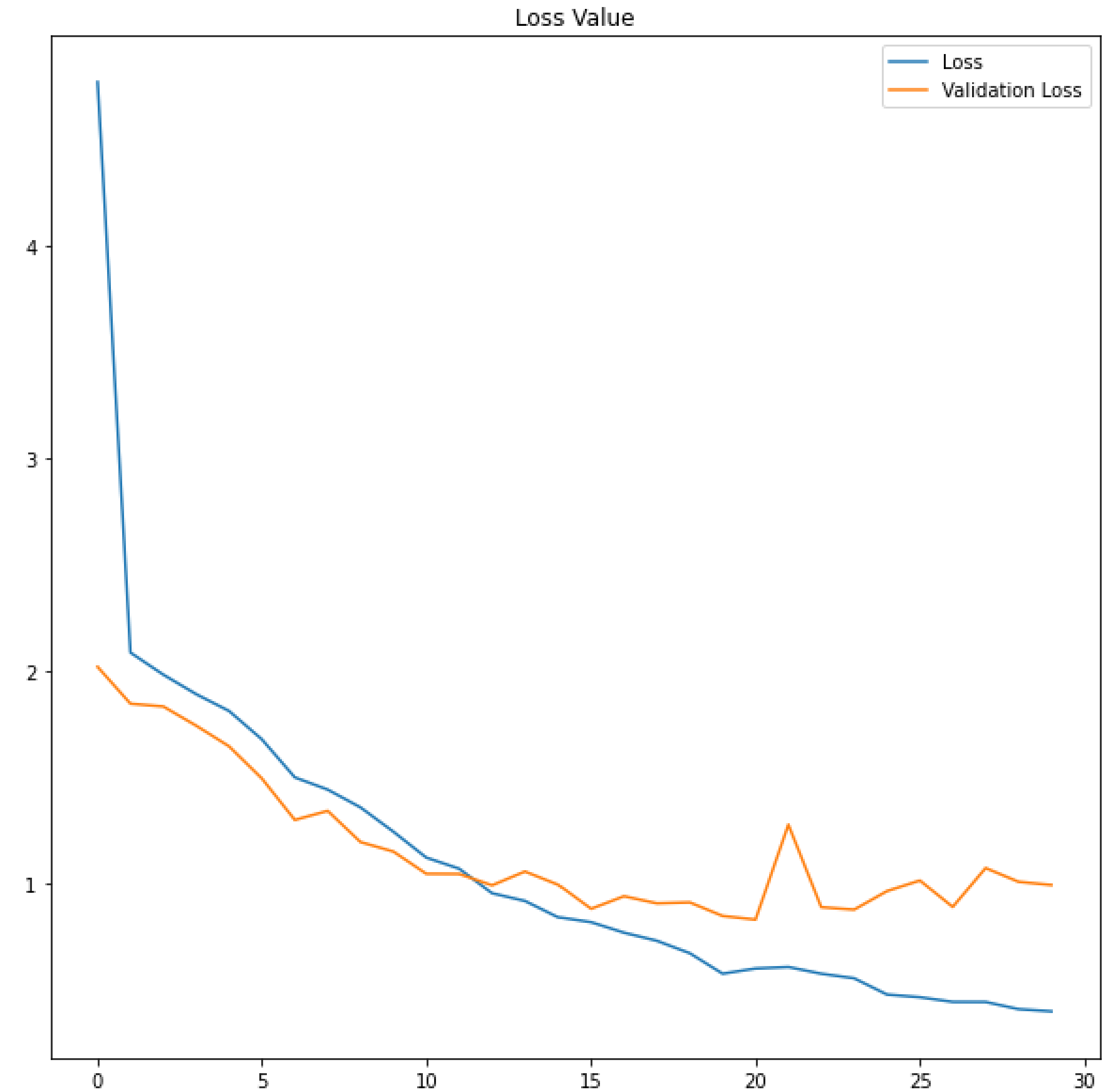
Below here are all the small changes and tuning that was done to the CNN model. This is not revelant to the Final Primary Model.

In [85]:
```
# Train/Validation/Test split
X_train, X_validation, y_train, Y_validation = mfcc_features_training_dataset,
mfcc_features_validation_dataset, one_hot_encoded_a, one_hot_encoded_b
X_test, y_test = mfcc_features_testing_dataset, one_hot_encoded_c

# Defining input shape for the neural network
input_shape = (X_train.shape[1], X_train.shape[2], 1)

# Reshape X_train and X_validation such that they are having the same shape as
the input shape
X_train = X_train.reshape(X_train.shape[0], X_train.shape[1], X_train.shape[2
], 1)
X_validation = X_validation.reshape(X_validation.shape[0], X_validation.shape[
1], X_validation.shape[2], 1)
X_test = X_test.reshape(X_test.shape[0], X_test.shape[1], X_test.shape[2], 1)

# Constructing the neural network architecture
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', strides=(1, 1),
    padding='same', input_shape=input_shape))
model.add(Conv2D(64, (3, 3), activation='relu', strides=(1, 1),
    padding='same'))
model.add(MaxPool2D((2, 2)))
model.add(Dropout(0.5))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(10, activation='softmax'))

model.compile(loss = 'categorical_crossentropy',
     optimizer = 'adam',
     metrics = ['acc'])

# Training the model
history = model.fit(X_train, y_train, epochs = 30, validation_data = (X_valida
tion, Y_validation))

# Displaying loss values
plt.figure(figsize = (10, 10))
plt.title('Loss Value')
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.legend(['Loss', 'Validation Loss'])
print('Loss:', history.history['loss'][-1])
print('Validation Loss:', history.history['val_loss'][-1])
plt.show()

# Displaying accuracy scores
plt.figure(figsize=(10, 10))
plt.title('Accuracy')
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.legend(['Accuracy', 'Validation Accuracy'])
print('Accuracy:', history.history['acc'][-1])
```

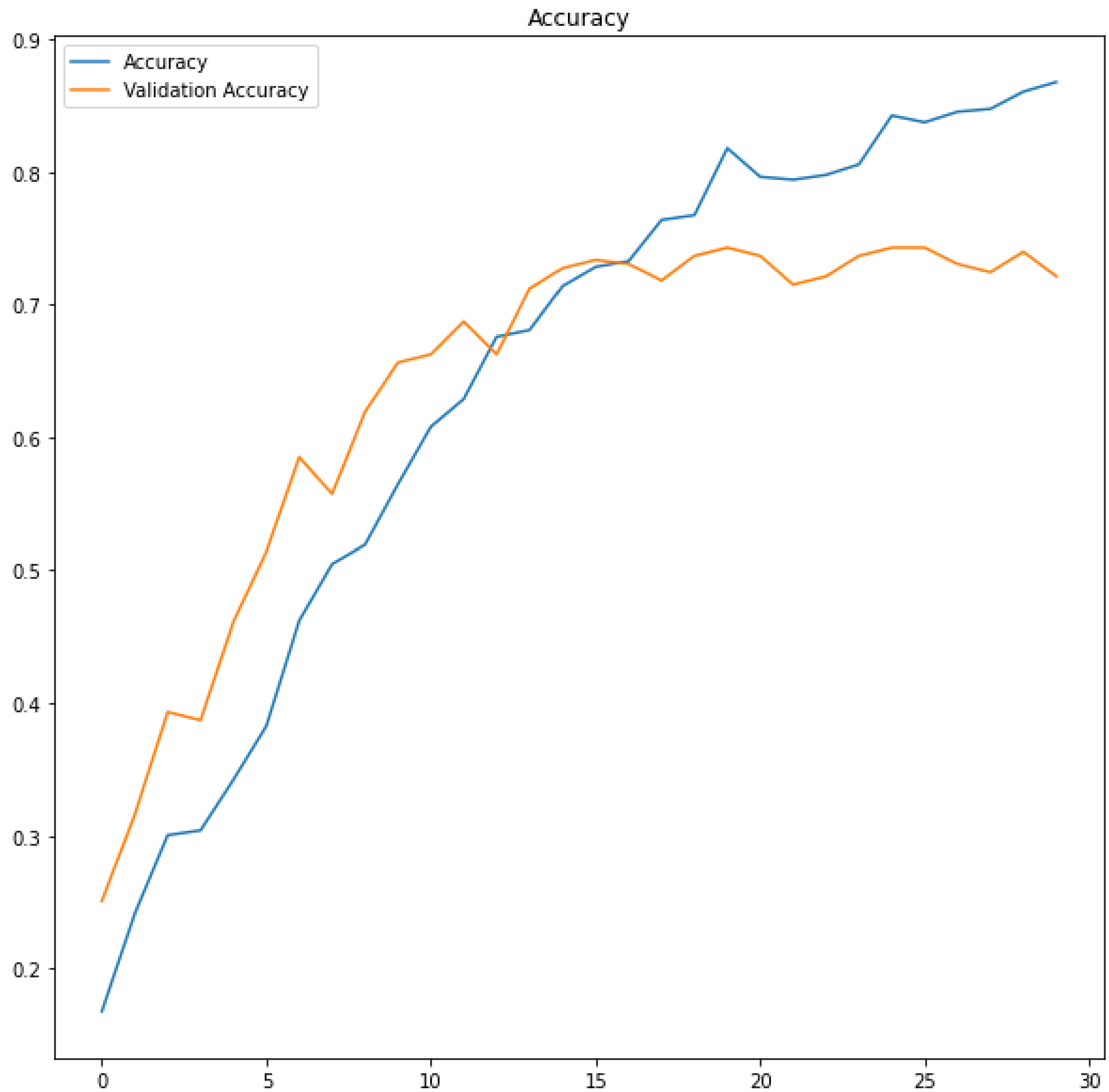
```
print('Validation Accuracy:', history.history['val_acc'][-1])
plt.show()

# Model evaluation
predictions = model.predict(X_validation)

predictions = np.argmax(predictions, axis=1)
Y_validation = one_hot_encoder_b.inverse_transform(Y_validation)

# Creating confusion matrix
cm = confusion_matrix(Y_validation, predictions)
plt.figure(figsize = (10, 10))
sns.heatmap(cm, annot = True, xticklabels = label_encoder_a.classes_, yticklab
els = label_encoder_b.classes_, fmt = 'd', cmap = plt.cm.Blues, cbar = False)
plt.xlabel('Predicted Label')
plt.ylabel('Actual Label')
plt.show()
```

```
Epoch 1/30
44/44 [==============================] - 22s 473ms/step - loss: 9.1289 - acc: 0.1317 - val_loss: 2.0204 - val_acc: 0.2508
Epoch 2/30
44/44 [==============================] - 20s 466ms/step - loss: 2.0949 - acc: 0.2403 - val_loss: 1.8470 - val_acc: 0.3158
Epoch 3/30
44/44 [==============================] - 20s 464ms/step - loss: 2.0196 - acc: 0.2725 - val_loss: 1.8339 - val_acc: 0.3932
Epoch 4/30
44/44 [==============================] - 20s 465ms/step - loss: 1.9043 - acc: 0.2940 - val_loss: 1.7436 - val_acc: 0.3870
Epoch 5/30
44/44 [==============================] - 21s 468ms/step - loss: 1.8563 - acc: 0.3334 - val_loss: 1.6465 - val_acc: 0.4613
Epoch 6/30
44/44 [==============================] - 20s 466ms/step - loss: 1.6951 - acc: 0.3760 - val_loss: 1.4949 - val_acc: 0.5139
Epoch 7/30
44/44 [==============================] - 20s 466ms/step - loss: 1.4623 - acc: 0.4732 - val_loss: 1.3004 - val_acc: 0.5851
Epoch 8/30
44/44 [==============================] - 21s 469ms/step - loss: 1.4355 - acc: 0.5012 - val_loss: 1.3429 - val_acc: 0.5573
Epoch 9/30
44/44 [==============================] - 20s 465ms/step - loss: 1.3672 - acc: 0.5156 - val_loss: 1.1962 - val_acc: 0.6192
Epoch 10/30
44/44 [==============================] - 21s 468ms/step - loss: 1.2506 - acc: 0.5633 - val_loss: 1.1515 - val_acc: 0.6563
Epoch 11/30
44/44 [==============================] - 21s 467ms/step - loss: 1.1826 - acc: 0.5978 - val_loss: 1.0468 - val_acc: 0.6625
Epoch 12/30
44/44 [==============================] - 21s 475ms/step - loss: 1.0814 - acc: 0.6189 - val_loss: 1.0461 - val_acc: 0.6873
Epoch 13/30
44/44 [==============================] - 20s 465ms/step - loss: 0.9825 - acc: 0.6621 - val_loss: 0.9928 - val_acc: 0.6625
Epoch 14/30
44/44 [==============================] - 21s 467ms/step - loss: 0.8491 - acc: 0.7095 - val_loss: 1.0577 - val_acc: 0.7121
Epoch 15/30
44/44 [==============================] - 20s 465ms/step - loss: 0.8339 - acc: 0.7123 - val_loss: 0.9961 - val_acc: 0.7276
Epoch 16/30
44/44 [==============================] - 20s 466ms/step - loss: 0.8772 - acc: 0.7203 - val_loss: 0.8824 - val_acc: 0.7337
Epoch 17/30
44/44 [==============================] - 21s 467ms/step - loss: 0.7713 - acc: 0.7378 - val_loss: 0.9415 - val_acc: 0.7307
Epoch 18/30
44/44 [==============================] - 21s 468ms/step - loss: 0.7096 - acc: 0.7721 - val_loss: 0.9080 - val_acc: 0.7183
Epoch 19/30
44/44 [==============================] - 20s 465ms/step - loss: 0.6422 - acc: 0.7721 - val_loss: 0.9125 - val_acc: 0.7368
```

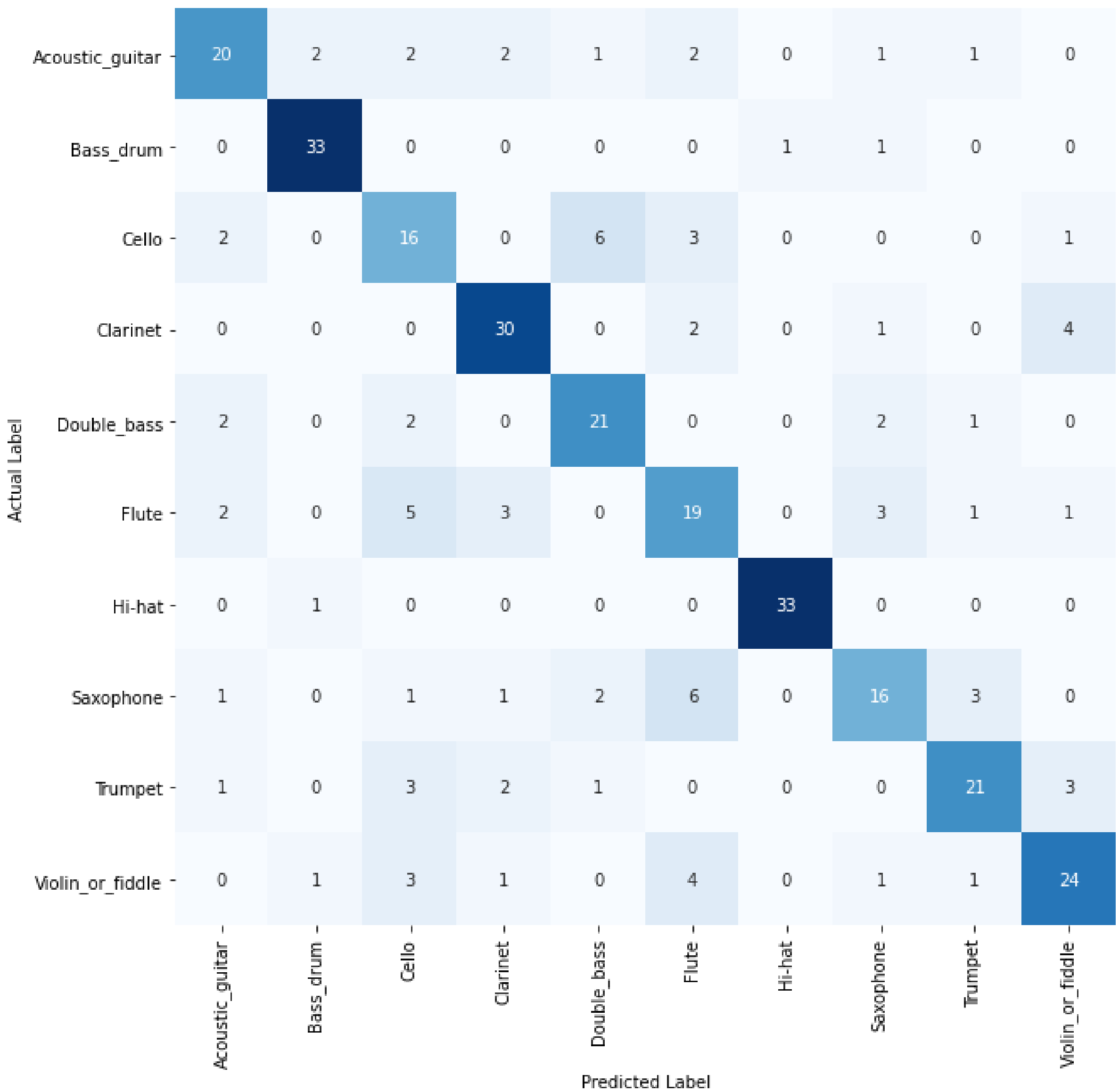
```
Epoch 20/30
44/44 [==============================] - 20s 466ms/step - loss: 0.6032 - acc:
0.8135 - val_loss: 0.8486 - val_acc: 0.7430
Epoch 21/30
44/44 [==============================] - 21s 473ms/step - loss: 0.5905 - acc:
0.8077 - val_loss: 0.8315 - val_acc: 0.7368
Epoch 22/30
44/44 [==============================] - 21s 468ms/step - loss: 0.5729 - acc:
0.8082 - val_loss: 1.2782 - val_acc: 0.7152
Epoch 23/30
44/44 [==============================] - 21s 471ms/step - loss: 0.5871 - acc:
0.8025 - val_loss: 0.8898 - val_acc: 0.7214
Epoch 24/30
44/44 [==============================] - 21s 487ms/step - loss: 0.5654 - acc:
0.7897 - val_loss: 0.8784 - val_acc: 0.7368
Epoch 25/30
44/44 [==============================] - 21s 470ms/step - loss: 0.5133 - acc:
0.8313 - val_loss: 0.9656 - val_acc: 0.7430
Epoch 26/30
44/44 [==============================] - 21s 466ms/step - loss: 0.4645 - acc:
0.8322 - val_loss: 1.0154 - val_acc: 0.7430
Epoch 27/30
44/44 [==============================] - 20s 464ms/step - loss: 0.4470 - acc:
0.8505 - val_loss: 0.8913 - val_acc: 0.7307
Epoch 28/30
44/44 [==============================] - 21s 466ms/step - loss: 0.4136 - acc:
0.8616 - val_loss: 1.0741 - val_acc: 0.7245
Epoch 29/30
44/44 [==============================] - 20s 463ms/step - loss: 0.4023 - acc:
0.8673 - val_loss: 1.0094 - val_acc: 0.7399
Epoch 30/30
44/44 [==============================] - 21s 470ms/step - loss: 0.3639 - acc:
0.8745 - val_loss: 0.9942 - val_acc: 0.7214
Loss: 0.3999413847923279
Validation Loss: 0.9941953420639038
```




```
Accuracy: 0.867870032787323
Validation Accuracy: 0.7213622331619263
```



Accuracy
Accuracy
Validation Accuracy
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0
5
10
15
20
25
30

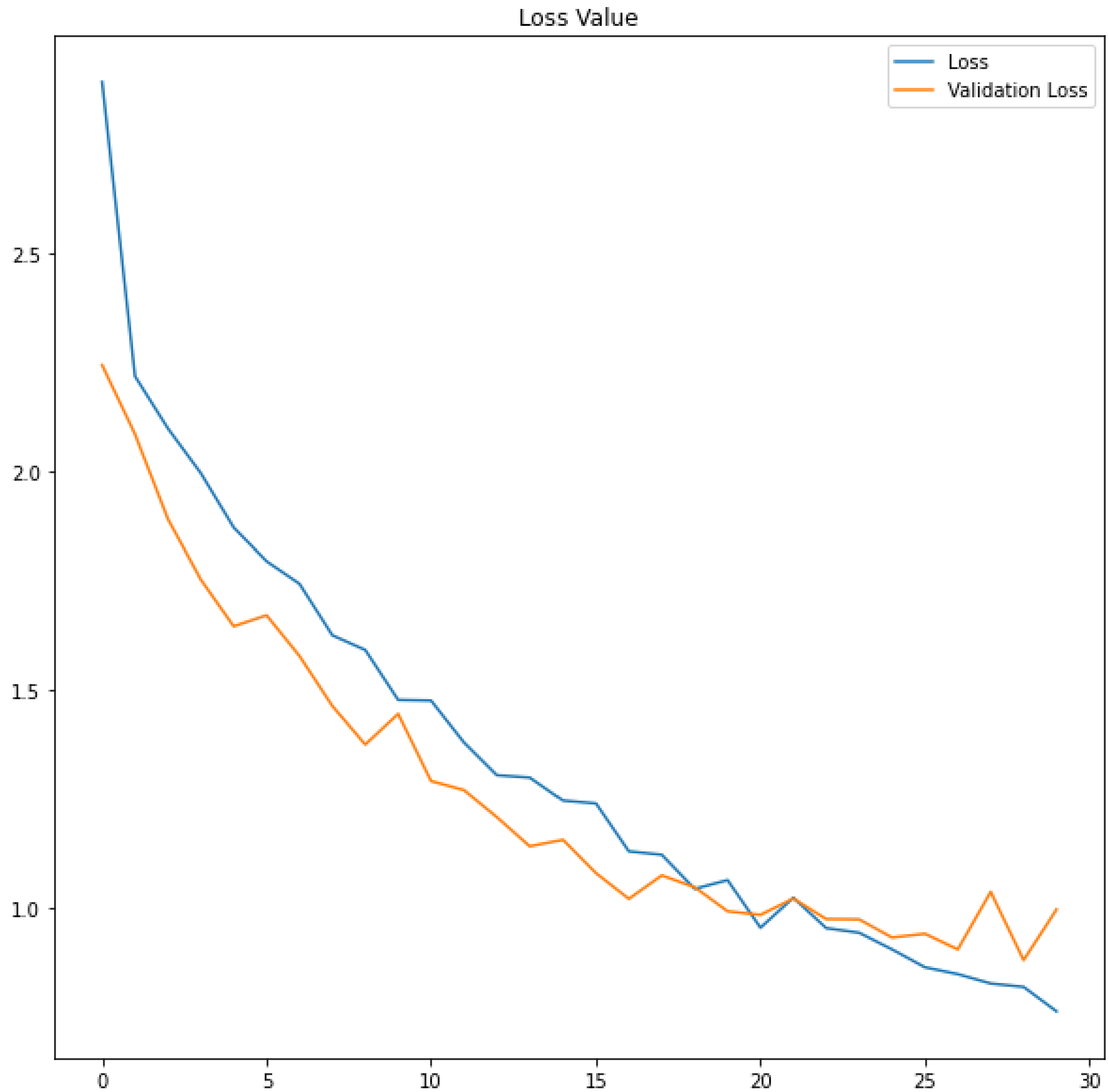

Actual Label
Acoustic_guitar 20 2 2 2 1 2 0 1 1 0
Bass_drum 0 33 0 0 0 0 1 1 0 0
Cello 2 0 16 0 6 3 0 0 0 1
Clarinet 0 0 0 30 0 2 0 1 0 4
Double_bass 2 0 2 0 21 0 0 2 1 0
Flute 2 0 5 3 0 19 0 3 1 1
Hi-hat 0 1 0 0 0 0 33 0 0 0
Saxophone 1 0 1 1 2 6 0 16 3 0
Trumpet 1 0 3 2 1 0 0 0 21 3
Violin_or_fiddle 0 1 3 1 0 4 0 1 1 24
Acoustic_guitar
Bass_drum
Cello
Clarinet
Double_bass
Flute
Hi-hat
Saxophone
Trumpet
Violin_or_fiddle
Predicted Label

In [86]:
```
# Train/Validation/Test split
X_train, X_validation, y_train, Y_validation = mfcc_features_training_dataset,
mfcc_features_validation_dataset, one_hot_encoded_a, one_hot_encoded_b
X_test, y_test = mfcc_features_testing_dataset, one_hot_encoded_c

# Defining input shape for the neural network
input_shape = (X_train.shape[1], X_train.shape[2], 1)

# Reshape X_train and X_validation such that they are having the same shape as
the input shape
X_train = X_train.reshape(X_train.shape[0], X_train.shape[1], X_train.shape[2
], 1)
X_validation = X_validation.reshape(X_validation.shape[0], X_validation.shape[
1], X_validation.shape[2], 1)
X_test = X_test.reshape(X_test.shape[0], X_test.shape[1], X_test.shape[2], 1)

# Constructing the neural network architecture
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', strides=(2, 2),
    padding='same', input_shape=input_shape))
model.add(Conv2D(64, (3, 3), activation='relu', strides=(2, 2),
    padding='same'))
model.add(MaxPool2D((2, 2)))
model.add(Dropout(0.5))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(10, activation='softmax'))

model.compile(loss = 'categorical_crossentropy',
     optimizer = 'adam',
     metrics = ['acc'])

# Training the model
history = model.fit(X_train, y_train, epochs = 30, validation_data = (X_valida
tion, Y_validation))

# Displaying loss values
plt.figure(figsize = (10, 10))
plt.title('Loss Value')
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.legend(['Loss', 'Validation Loss'])
print('Loss:', history.history['loss'][-1])
print('Validation Loss:', history.history['val_loss'][-1])
plt.show()

# Displaying accuracy scores
plt.figure(figsize=(10, 10))
plt.title('Accuracy')
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.legend(['Accuracy', 'Validation Accuracy'])
print('Accuracy:', history.history['acc'][-1])
```

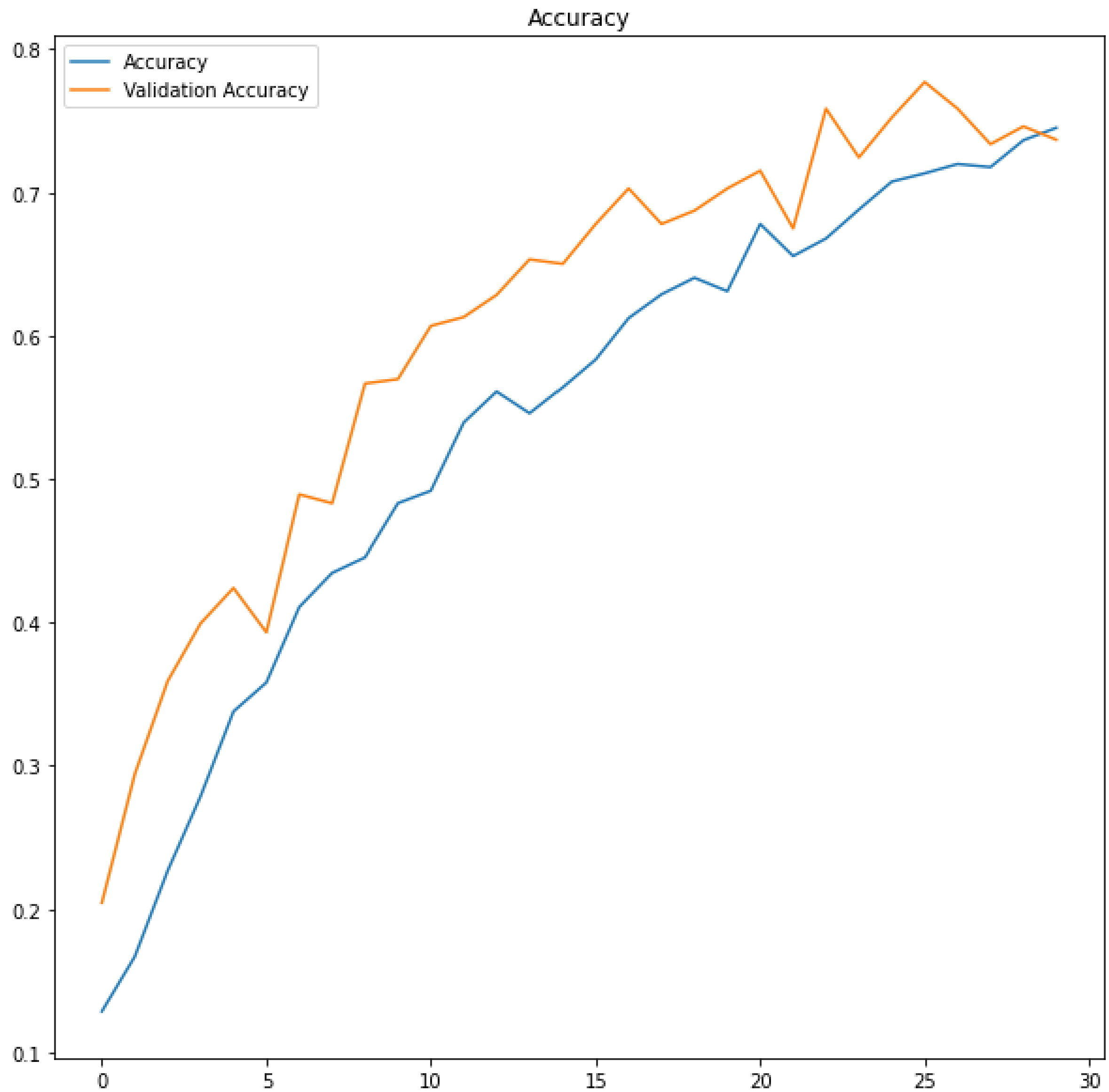
```
print('Validation Accuracy:', history.history['val_acc'][-1])
plt.show()

# Model evaluation
predictions = model.predict(X_validation)

predictions = np.argmax(predictions, axis=1)
Y_validation = one_hot_encoder_b.inverse_transform(Y_validation)

# Creating confusion matrix
cm = confusion_matrix(Y_validation, predictions)
plt.figure(figsize = (10, 10))
sns.heatmap(cm, annot = True, xticklabels = label_encoder_a.classes_, yticklab
els = label_encoder_b.classes_, fmt = 'd', cmap = plt.cm.Blues, cbar = False)
plt.xlabel('Predicted Label')
plt.ylabel('Actual Label')
plt.show()
```

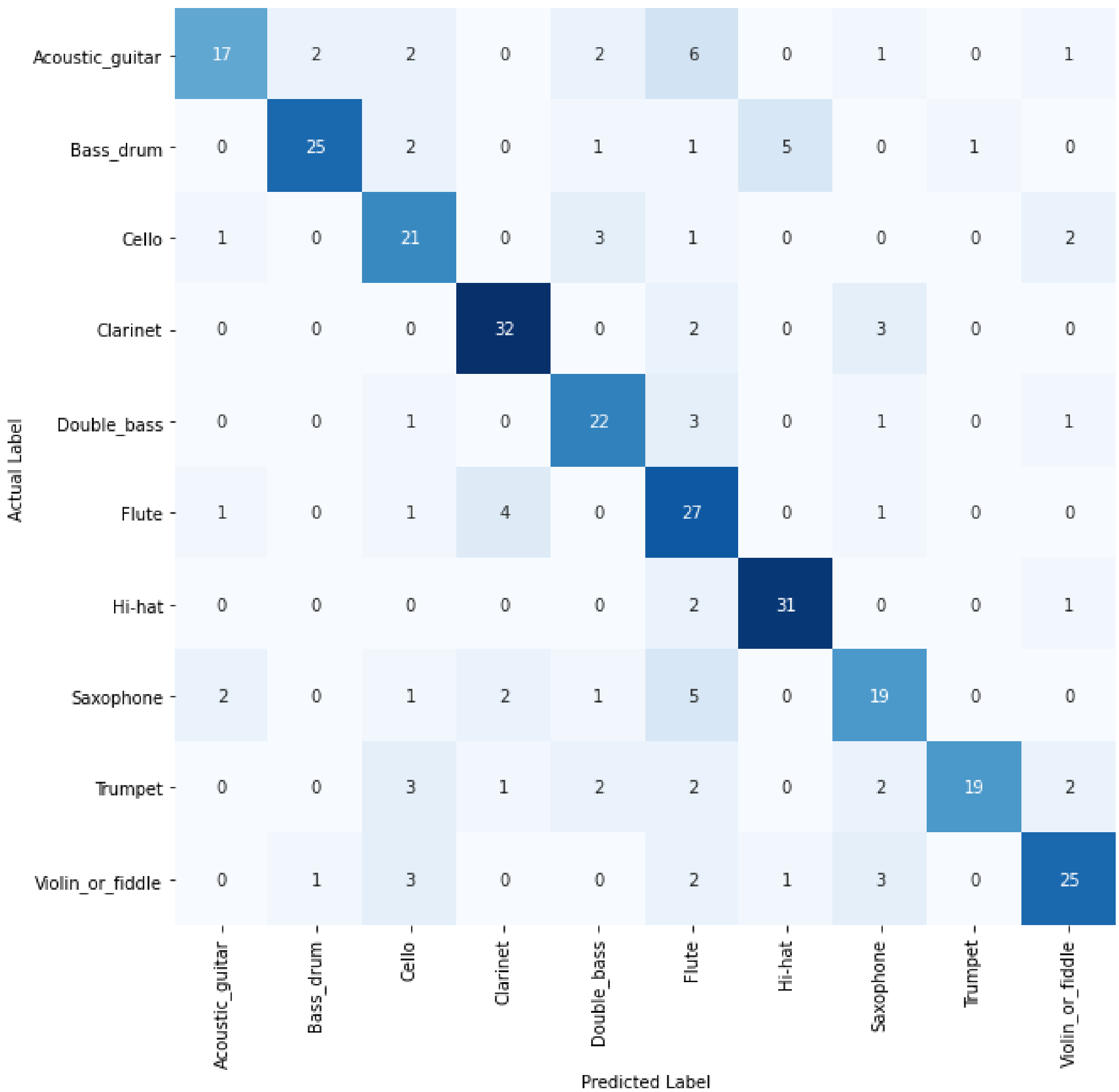
```
Epoch 1/30
44/44 [==============================] - 3s 62ms/step - loss: 4.0524 - acc:
0.1185 - val_loss: 2.2430 - val_acc: 0.2043
Epoch 2/30
44/44 [==============================] - 2s 56ms/step - loss: 2.2519 - acc:
0.1397 - val_loss: 2.0844 - val_acc: 0.2941
Epoch 3/30
44/44 [==============================] - 2s 56ms/step - loss: 2.1479 - acc:
0.2097 - val_loss: 1.8904 - val_acc: 0.3591
Epoch 4/30
44/44 [==============================] - 2s 57ms/step - loss: 2.0190 - acc:
0.2759 - val_loss: 1.7526 - val_acc: 0.3994
Epoch 5/30
44/44 [==============================] - 3s 57ms/step - loss: 1.9021 - acc:
0.3313 - val_loss: 1.6456 - val_acc: 0.4241
Epoch 6/30
44/44 [==============================] - 2s 56ms/step - loss: 1.8357 - acc:
0.3408 - val_loss: 1.6707 - val_acc: 0.3932
Epoch 7/30
44/44 [==============================] - 2s 57ms/step - loss: 1.7481 - acc:
0.3942 - val_loss: 1.5776 - val_acc: 0.4892
Epoch 8/30
44/44 [==============================] - 3s 57ms/step - loss: 1.6353 - acc:
0.4224 - val_loss: 1.4630 - val_acc: 0.4830
Epoch 9/30
44/44 [==============================] - 3s 59ms/step - loss: 1.6073 - acc:
0.4325 - val_loss: 1.3747 - val_acc: 0.5666
Epoch 10/30
44/44 [==============================] - 3s 58ms/step - loss: 1.4128 - acc:
0.5043 - val_loss: 1.4450 - val_acc: 0.5697
Epoch 11/30
44/44 [==============================] - 3s 58ms/step - loss: 1.5515 - acc:
0.4744 - val_loss: 1.2911 - val_acc: 0.6068
Epoch 12/30
44/44 [==============================] - 2s 56ms/step - loss: 1.4023 - acc:
0.5315 - val_loss: 1.2704 - val_acc: 0.6130
Epoch 13/30
44/44 [==============================] - 3s 57ms/step - loss: 1.2979 - acc:
0.5615 - val_loss: 1.2085 - val_acc: 0.6285
Epoch 14/30
44/44 [==============================] - 2s 57ms/step - loss: 1.2560 - acc:
0.5679 - val_loss: 1.1417 - val_acc: 0.6533
Epoch 15/30
44/44 [==============================] - 2s 56ms/step - loss: 1.2107 - acc:
0.5772 - val_loss: 1.1564 - val_acc: 0.6502
Epoch 16/30
44/44 [==============================] - 2s 56ms/step - loss: 1.2766 - acc:
0.5837 - val_loss: 1.0802 - val_acc: 0.6780
Epoch 17/30
44/44 [==============================] - 2s 56ms/step - loss: 1.2068 - acc:
0.5934 - val_loss: 1.0213 - val_acc: 0.7028
Epoch 18/30
44/44 [==============================] - 2s 55ms/step - loss: 1.0560 - acc:
0.6356 - val_loss: 1.0750 - val_acc: 0.6780
Epoch 19/30
44/44 [==============================] - 2s 57ms/step - loss: 1.0179 - acc:
0.6461 - val_loss: 1.0481 - val_acc: 0.6873
```

```
Epoch 20/30
44/44 [==============================] - 2s 56ms/step - loss: 1.0322 - acc:
0.6382 - val_loss: 0.9928 - val_acc: 0.7028
Epoch 21/30
44/44 [==============================] - 2s 56ms/step - loss: 1.0123 - acc:
0.6693 - val_loss: 0.9847 - val_acc: 0.7152
Epoch 22/30
44/44 [==============================] - 2s 56ms/step - loss: 0.9710 - acc:
0.6614 - val_loss: 1.0218 - val_acc: 0.6749
Epoch 23/30
44/44 [==============================] - 2s 55ms/step - loss: 0.9607 - acc:
0.6604 - val_loss: 0.9750 - val_acc: 0.7585
Epoch 24/30
44/44 [==============================] - 2s 56ms/step - loss: 0.9248 - acc:
0.6950 - val_loss: 0.9743 - val_acc: 0.7245
Epoch 25/30
44/44 [==============================] - 2s 55ms/step - loss: 0.8728 - acc:
0.6978 - val_loss: 0.9327 - val_acc: 0.7523
Epoch 26/30
44/44 [==============================] - 2s 56ms/step - loss: 0.8716 - acc:
0.7175 - val_loss: 0.9412 - val_acc: 0.7771
Epoch 27/30
44/44 [==============================] - 2s 55ms/step - loss: 0.8214 - acc:
0.7225 - val_loss: 0.9055 - val_acc: 0.7585
Epoch 28/30
44/44 [==============================] - 2s 57ms/step - loss: 0.8395 - acc:
0.7101 - val_loss: 1.0373 - val_acc: 0.7337
Epoch 29/30
44/44 [==============================] - 2s 57ms/step - loss: 0.8160 - acc:
0.7322 - val_loss: 0.8807 - val_acc: 0.7461
Epoch 30/30
44/44 [==============================] - 2s 56ms/step - loss: 0.7482 - acc:
0.7584 - val_loss: 0.9969 - val_acc: 0.7368
Loss: 0.7635603547096252
Validation Loss: 0.9968526363372803
```




```
Accuracy: 0.7451263666152954
Validation Accuracy: 0.7368420958518982
```


Accuracy
Accuracy
Validation Accuracy
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0
5
10
15
20
25
30



Acoustic_guitar
Bass_drum
Cello
Clarinet
Double_bass
Flute
Hi-hat
Saxophone
Trumpet
Violin_or_fiddle
Actual Label
17 2 2 0 2 6 0 1 0 1
0 25 2 0 1 1 5 0 1 0
1 0 21 0 3 1 0 0 0 2
0 0 0 32 0 2 0 3 0 0
0 0 1 0 22 3 0 1 0 1
1 0 1 4 0 27 0 1 0 0
0 0 0 0 0 2 31 0 0 1
2 0 1 2 1 5 0 19 0 0
0 0 3 1 2 2 0 2 19 2
0 1 3 0 0 2 1 3 0 25
Acoustic_guitar
Bass_drum
Cello
Clarinet
Double_bass
Flute
Hi-hat
Saxophone
Trumpet
Violin_or_fiddle
Predicted Label

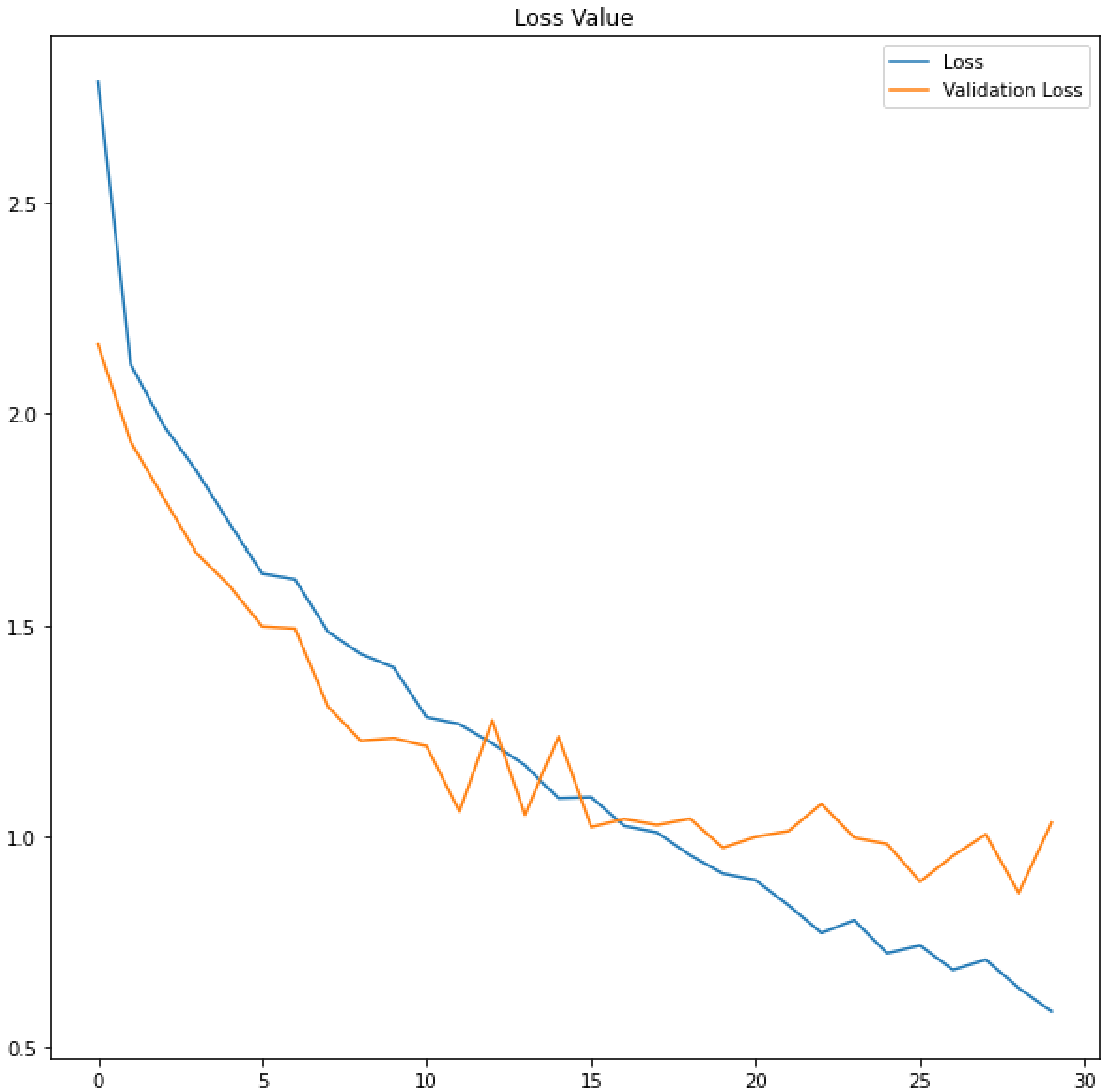
In [87]:

```
# Train/Validation/Test split
X_train, X_validation, y_train, Y_validation = mfcc_features_training_dataset,
mfcc_features_validation_dataset, one_hot_encoded_a, one_hot_encoded_b
X_test, y_test = mfcc_features_testing_dataset, one_hot_encoded_c

# Defining input shape for the neural network
input_shape = (X_train.shape[1], X_train.shape[2], 1)

# Reshape X_train and X_validation such that they are having the same shape as
the input shape
X_train = X_train.reshape(X_train.shape[0], X_train.shape[1], X_train.shape[2
], 1)
X_validation = X_validation.reshape(X_validation.shape[0], X_validation.shape[
1], X_validation.shape[2], 1)
X_test = X_test.reshape(X_test.shape[0], X_test.shape[1], X_test.shape[2], 1)

# Constructing the neural network architecture
model = Sequential()
model.add(Conv2D(32, (4, 4), activation='relu', strides=(2, 2),
    padding='same', input_shape=input_shape))
model.add(Conv2D(64, (4, 4), activation='relu', strides=(2, 2),
    padding='same'))
model.add(MaxPool2D((2, 2)))
model.add(Dropout(0.5))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(10, activation='softmax'))

model.compile(loss = 'categorical_crossentropy',
     optimizer = 'adam',
     metrics = ['acc'])

# Training the model
history = model.fit(X_train, y_train, epochs = 30, validation_data = (X_valida
tion, Y_validation))

# Displaying loss values
plt.figure(figsize = (10, 10))
plt.title('Loss Value')
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.legend(['Loss', 'Validation Loss'])
print('Loss:', history.history['loss'][-1])
print('Validation Loss:', history.history['val_loss'][-1])
plt.show()

# Displaying accuracy scores
plt.figure(figsize=(10, 10))
plt.title('Accuracy')
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.legend(['Accuracy', 'Validation Accuracy'])
print('Accuracy:', history.history['acc'][-1])
```

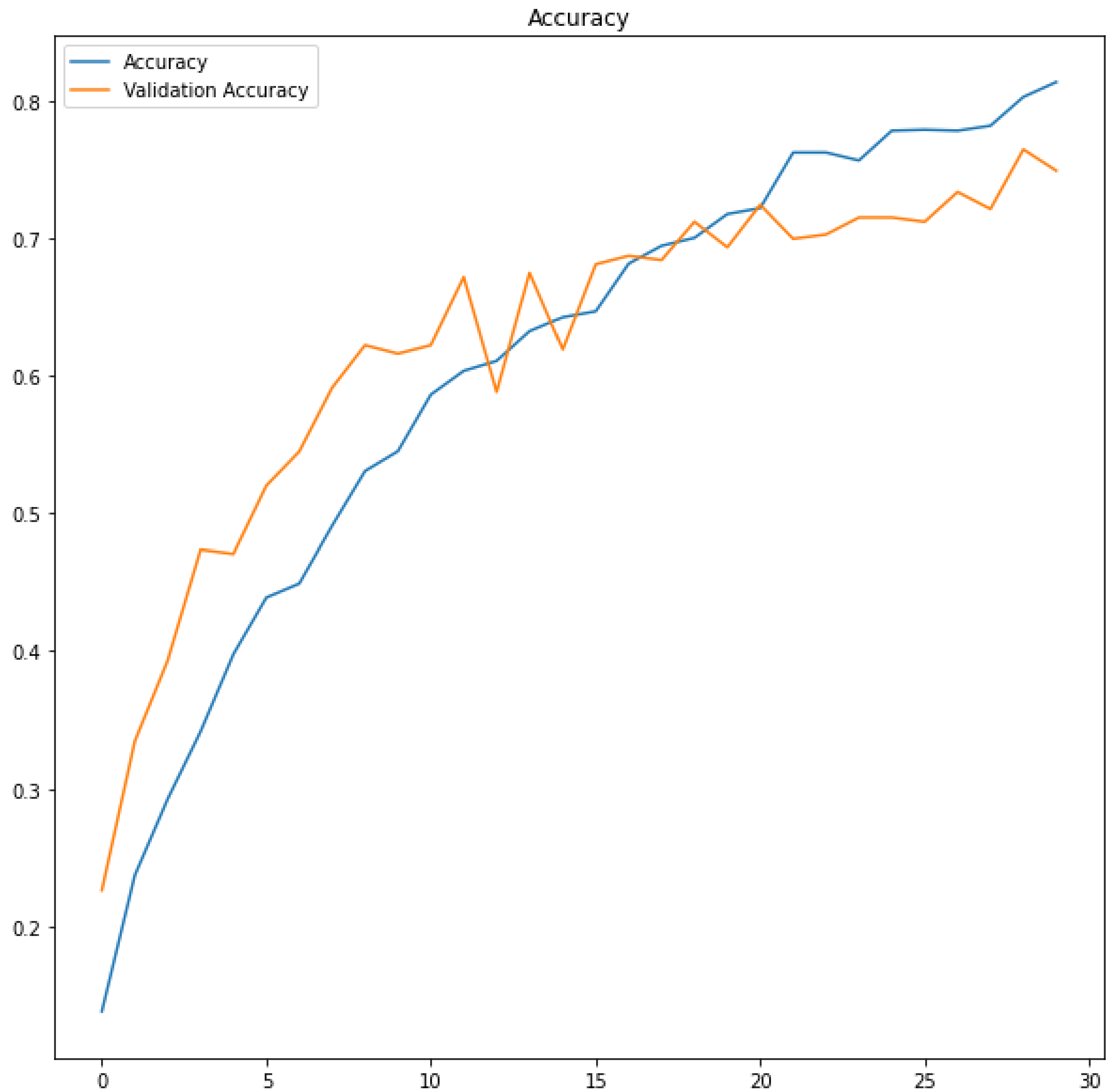
```
print('Validation Accuracy:', history.history['val_acc'][-1])
plt.show()

# Model evaluation
predictions = model.predict(X_validation)

predictions = np.argmax(predictions, axis=1)
Y_validation = one_hot_encoder_b.inverse_transform(Y_validation)

# Creating confusion matrix
cm = confusion_matrix(Y_validation, predictions)
plt.figure(figsize = (10, 10))
sns.heatmap(cm, annot = True, xticklabels = label_encoder_a.classes_, yticklab
els = label_encoder_b.classes_, fmt = 'd', cmap = plt.cm.Blues, cbar = False)
plt.xlabel('Predicted Label')
plt.ylabel('Actual Label')
plt.show()
```

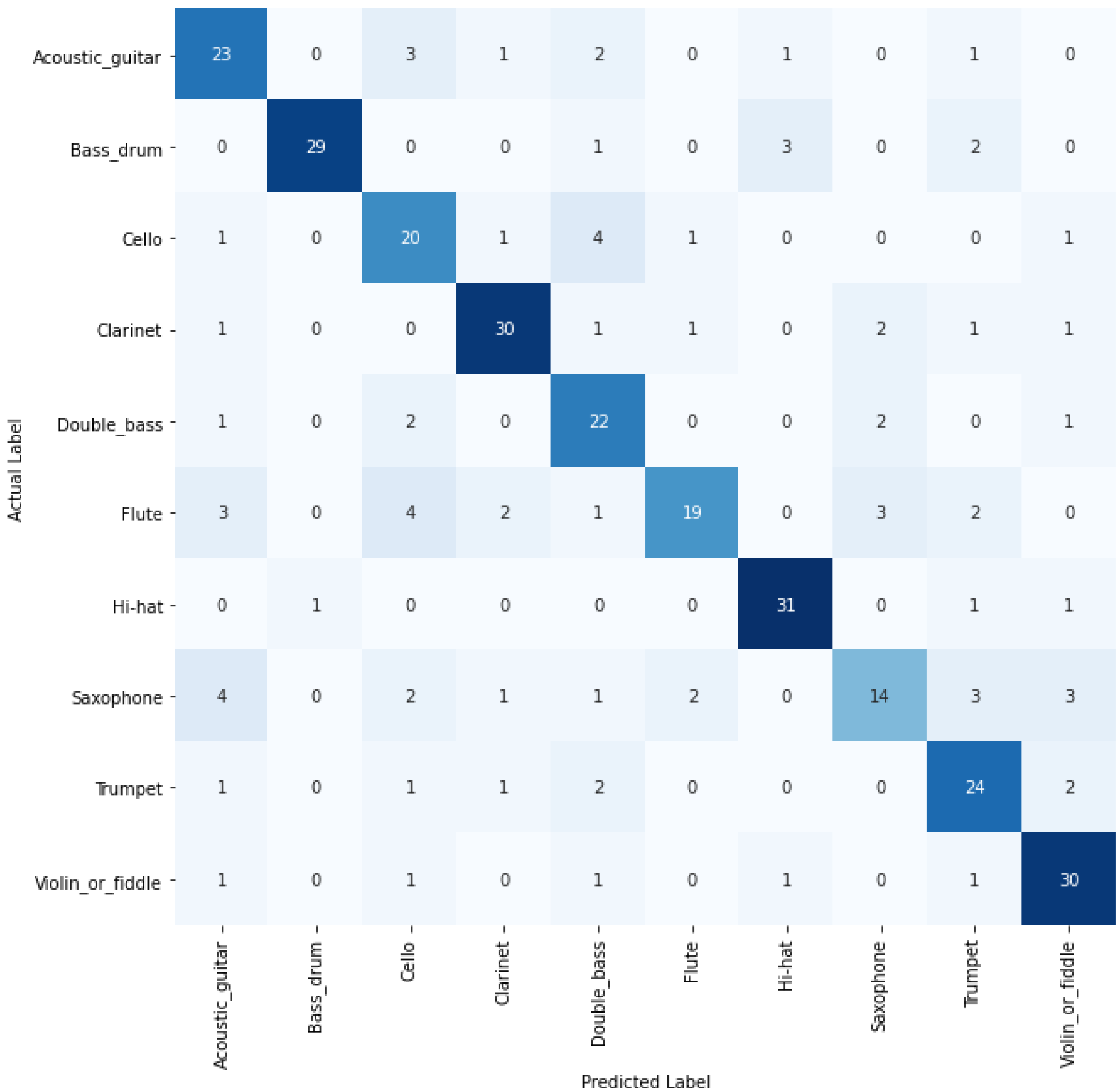
```
Epoch 1/30
44/44 [==============================] - 4s 88ms/step - loss: 3.7622 - acc:
0.1285 - val_loss: 2.1625 - val_acc: 0.2260
Epoch 2/30
44/44 [==============================] - 4s 83ms/step - loss: 2.1664 - acc:
0.2039 - val_loss: 1.9322 - val_acc: 0.3344
Epoch 3/30
44/44 [==============================] - 4s 82ms/step - loss: 1.9960 - acc:
0.2792 - val_loss: 1.7993 - val_acc: 0.3932
Epoch 4/30
44/44 [==============================] - 4s 80ms/step - loss: 1.8599 - acc:
0.3329 - val_loss: 1.6696 - val_acc: 0.4737
Epoch 5/30
44/44 [==============================] - 4s 80ms/step - loss: 1.7352 - acc:
0.3949 - val_loss: 1.5936 - val_acc: 0.4706
Epoch 6/30
44/44 [==============================] - 4s 80ms/step - loss: 1.6292 - acc:
0.4394 - val_loss: 1.4965 - val_acc: 0.5201
Epoch 7/30
44/44 [==============================] - 4s 82ms/step - loss: 1.6122 - acc:
0.4576 - val_loss: 1.4915 - val_acc: 0.5449
Epoch 8/30
44/44 [==============================] - 4s 81ms/step - loss: 1.5428 - acc:
0.4588 - val_loss: 1.3070 - val_acc: 0.5913
Epoch 9/30
44/44 [==============================] - 4s 82ms/step - loss: 1.3609 - acc:
0.5548 - val_loss: 1.2260 - val_acc: 0.6223
Epoch 10/30
44/44 [==============================] - 4s 82ms/step - loss: 1.3479 - acc:
0.5480 - val_loss: 1.2325 - val_acc: 0.6161
Epoch 11/30
44/44 [==============================] - 4s 82ms/step - loss: 1.2537 - acc:
0.5899 - val_loss: 1.2135 - val_acc: 0.6223
Epoch 12/30
44/44 [==============================] - 4s 81ms/step - loss: 1.2675 - acc:
0.6104 - val_loss: 1.0589 - val_acc: 0.6718
Epoch 13/30
44/44 [==============================] - 4s 82ms/step - loss: 1.1934 - acc:
0.6251 - val_loss: 1.2743 - val_acc: 0.5882
Epoch 14/30
44/44 [==============================] - 4s 81ms/step - loss: 1.1460 - acc:
0.6350 - val_loss: 1.0506 - val_acc: 0.6749
Epoch 15/30
44/44 [==============================] - 4s 81ms/step - loss: 1.0757 - acc:
0.6391 - val_loss: 1.2359 - val_acc: 0.6192
Epoch 16/30
44/44 [==============================] - 4s 82ms/step - loss: 1.1035 - acc:
0.6431 - val_loss: 1.0222 - val_acc: 0.6811
Epoch 17/30
44/44 [==============================] - 4s 82ms/step - loss: 0.9851 - acc:
0.6917 - val_loss: 1.0414 - val_acc: 0.6873
Epoch 18/30
44/44 [==============================] - 4s 82ms/step - loss: 0.9541 - acc:
0.7075 - val_loss: 1.0269 - val_acc: 0.6842
Epoch 19/30
44/44 [==============================] - 4s 82ms/step - loss: 0.9211 - acc:
0.7105 - val_loss: 1.0420 - val_acc: 0.7121
```

```
Epoch 20/30
44/44 [==============================] - 4s 82ms/step - loss: 1.0328 - acc: 0.6884 - val_loss: 0.9734 - val_acc: 0.6935
Epoch 21/30
44/44 [==============================] - 4s 82ms/step - loss: 0.9727 - acc: 0.6848 - val_loss: 0.9990 - val_acc: 0.7245
Epoch 22/30
44/44 [==============================] - 4s 82ms/step - loss: 0.8614 - acc: 0.7602 - val_loss: 1.0127 - val_acc: 0.6997
Epoch 23/30
44/44 [==============================] - 4s 82ms/step - loss: 0.7639 - acc: 0.7736 - val_loss: 1.0770 - val_acc: 0.7028
Epoch 24/30
44/44 [==============================] - 4s 83ms/step - loss: 0.7836 - acc: 0.7581 - val_loss: 0.9972 - val_acc: 0.7152
Epoch 25/30
44/44 [==============================] - 4s 84ms/step - loss: 0.6951 - acc: 0.7687 - val_loss: 0.9821 - val_acc: 0.7152
Epoch 26/30
44/44 [==============================] - 4s 81ms/step - loss: 0.7556 - acc: 0.7779 - val_loss: 0.8931 - val_acc: 0.7121
Epoch 27/30
44/44 [==============================] - 4s 82ms/step - loss: 0.7151 - acc: 0.7804 - val_loss: 0.9542 - val_acc: 0.7337
Epoch 28/30
44/44 [==============================] - 4s 82ms/step - loss: 0.6742 - acc: 0.7861 - val_loss: 1.0049 - val_acc: 0.7214
Epoch 29/30
44/44 [==============================] - 4s 82ms/step - loss: 0.6756 - acc: 0.8036 - val_loss: 0.8662 - val_acc: 0.7647
Epoch 30/30
44/44 [==============================] - 4s 82ms/step - loss: 0.6124 - acc: 0.8054 - val_loss: 1.0325 - val_acc: 0.7492
Loss: 0.5862382650375366
Validation Loss: 1.0325456857681274
```




```
Accuracy: 0.8137184381484985
Validation Accuracy: 0.7492260336875916
```


Accuracy
Accuracy
Validation Accuracy
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0
5
10
15
20
25
30



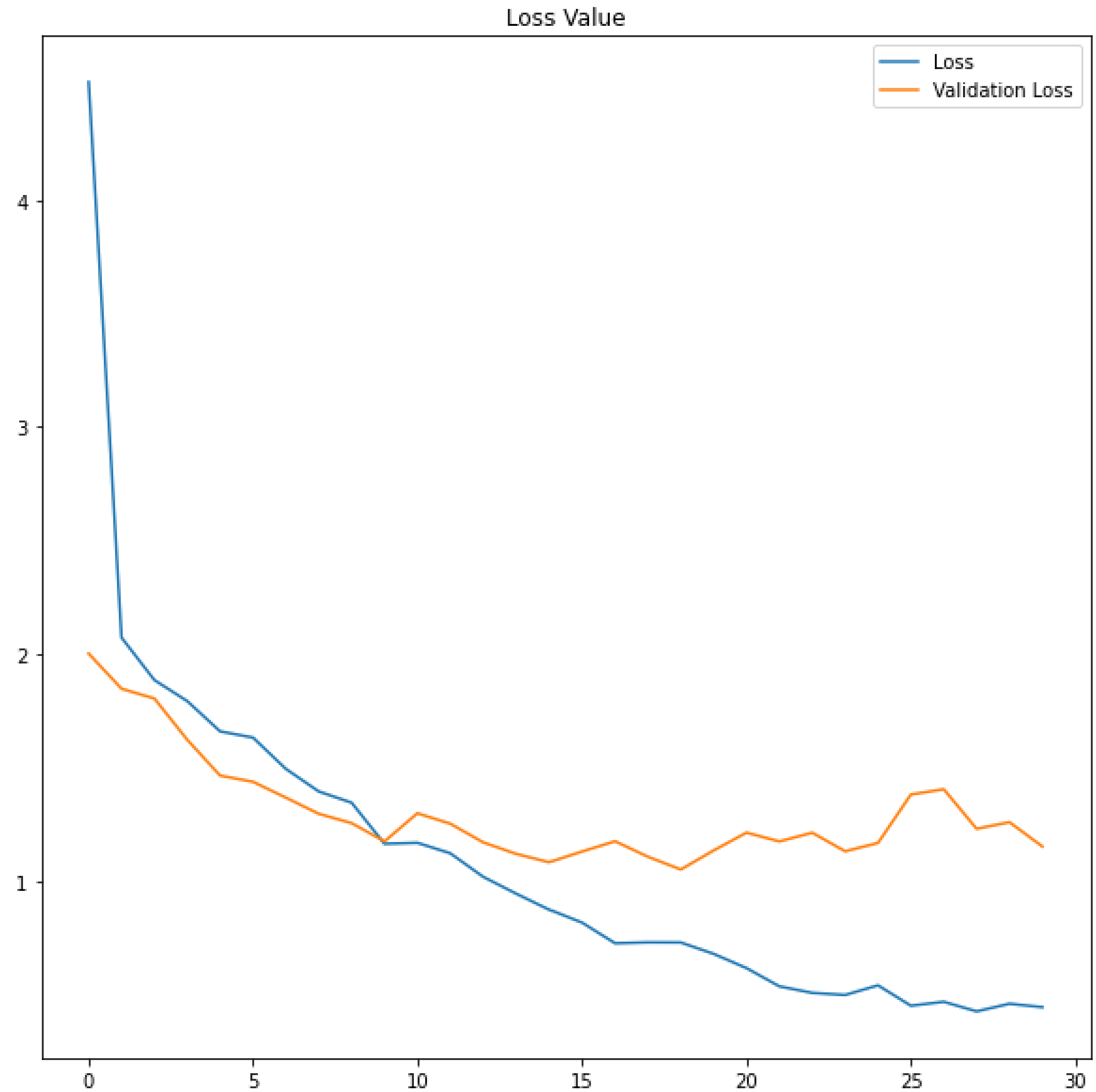
Actual Label
Predicted Label
Acoustic_guitar
Bass_drum
Cello
Clarinet
Double_bass
Flute
Hi-hat
Saxophone
Trumpet
Violin_or_fiddle
23 0 3 1 2 0 1 0 1 0
0 29 0 0 1 0 3 0 2 0
1 0 20 1 4 1 0 0 0 1
1 0 0 30 1 1 0 2 1 1
1 0 2 0 22 0 0 2 0 1
3 0 4 2 1 19 0 3 2 0
0 1 0 0 0 0 31 0 1 1
4 0 2 1 1 2 0 14 3 3
1 0 1 1 2 0 0 0 24 2
1 0 1 0 1 0 1 0 1 30

In [88]:
```
# Train/Validation/Test split
X_train, X_validation, y_train, Y_validation = mfcc_features_training_dataset,
mfcc_features_validation_dataset, one_hot_encoded_a, one_hot_encoded_b
X_test, y_test = mfcc_features_testing_dataset, one_hot_encoded_c

# Defining input shape for the neural network
input_shape = (X_train.shape[1], X_train.shape[2], 1)

# Reshape X_train and X_validation such that they are having the same shape as
the input shape
X_train = X_train.reshape(X_train.shape[0], X_train.shape[1], X_train.shape[2
], 1)
X_validation = X_validation.reshape(X_validation.shape[0], X_validation.shape[
1], X_validation.shape[2], 1)
X_test = X_test.reshape(X_test.shape[0], X_test.shape[1], X_test.shape[2], 1)

# Constructing the neural network architecture
model = Sequential()
model.add(Conv2D(32, (4, 4), activation='relu', strides=(1, 1),
    padding='same', input_shape=input_shape))
model.add(Conv2D(64, (4, 4), activation='relu', strides=(1, 1),
    padding='same'))
model.add(MaxPool2D((2, 2)))
model.add(Dropout(0.5))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(10, activation='softmax'))

model.compile(loss = 'categorical_crossentropy',
     optimizer = 'adam',
     metrics = ['acc'])

# Training the model
history = model.fit(X_train, y_train, epochs = 30, validation_data = (X_valida
tion, Y_validation))

# Displaying loss values
plt.figure(figsize = (10, 10))
plt.title('Loss Value')
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.legend(['Loss', 'Validation Loss'])
print('Loss:', history.history['loss'][-1])
print('Validation Loss:', history.history['val_loss'][-1])
plt.show()

# Displaying accuracy scores
plt.figure(figsize=(10, 10))
plt.title('Accuracy')
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.legend(['Accuracy', 'Validation Accuracy'])
print('Accuracy:', history.history['acc'][-1])
```

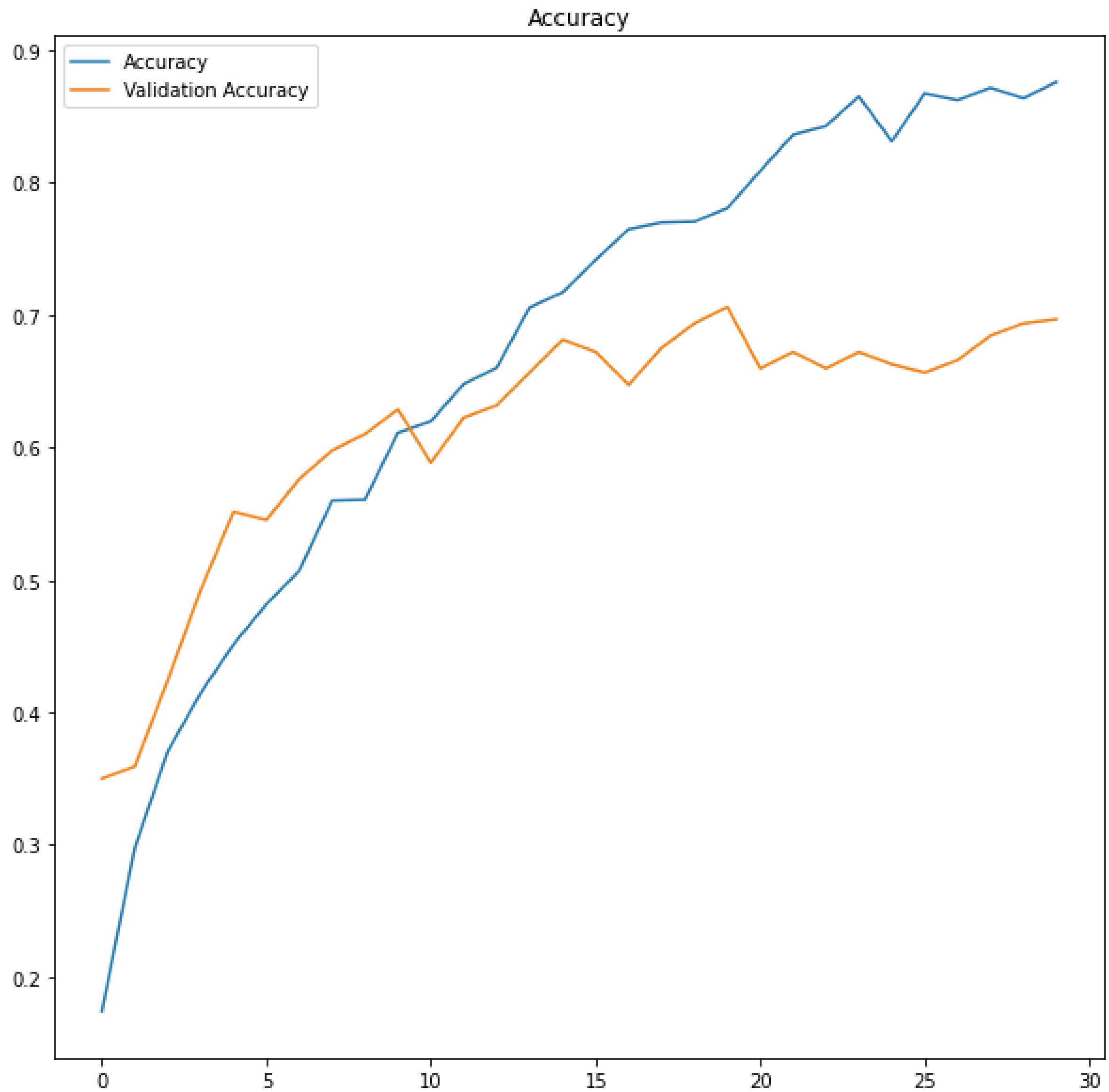
```
print('Validation Accuracy:', history.history['val_acc'][-1])
plt.show()

# Model evaluation
predictions = model.predict(X_validation)

predictions = np.argmax(predictions, axis=1)
Y_validation = one_hot_encoder_b.inverse_transform(Y_validation)

# Creating confusion matrix
cm = confusion_matrix(Y_validation, predictions)
plt.figure(figsize = (10, 10))
sns.heatmap(cm, annot = True, xticklabels = label_encoder_a.classes_, yticklab
els = label_encoder_b.classes_, fmt = 'd', cmap = plt.cm.Blues, cbar = False)
plt.xlabel('Predicted Label')
plt.ylabel('Actual Label')
plt.show()
```

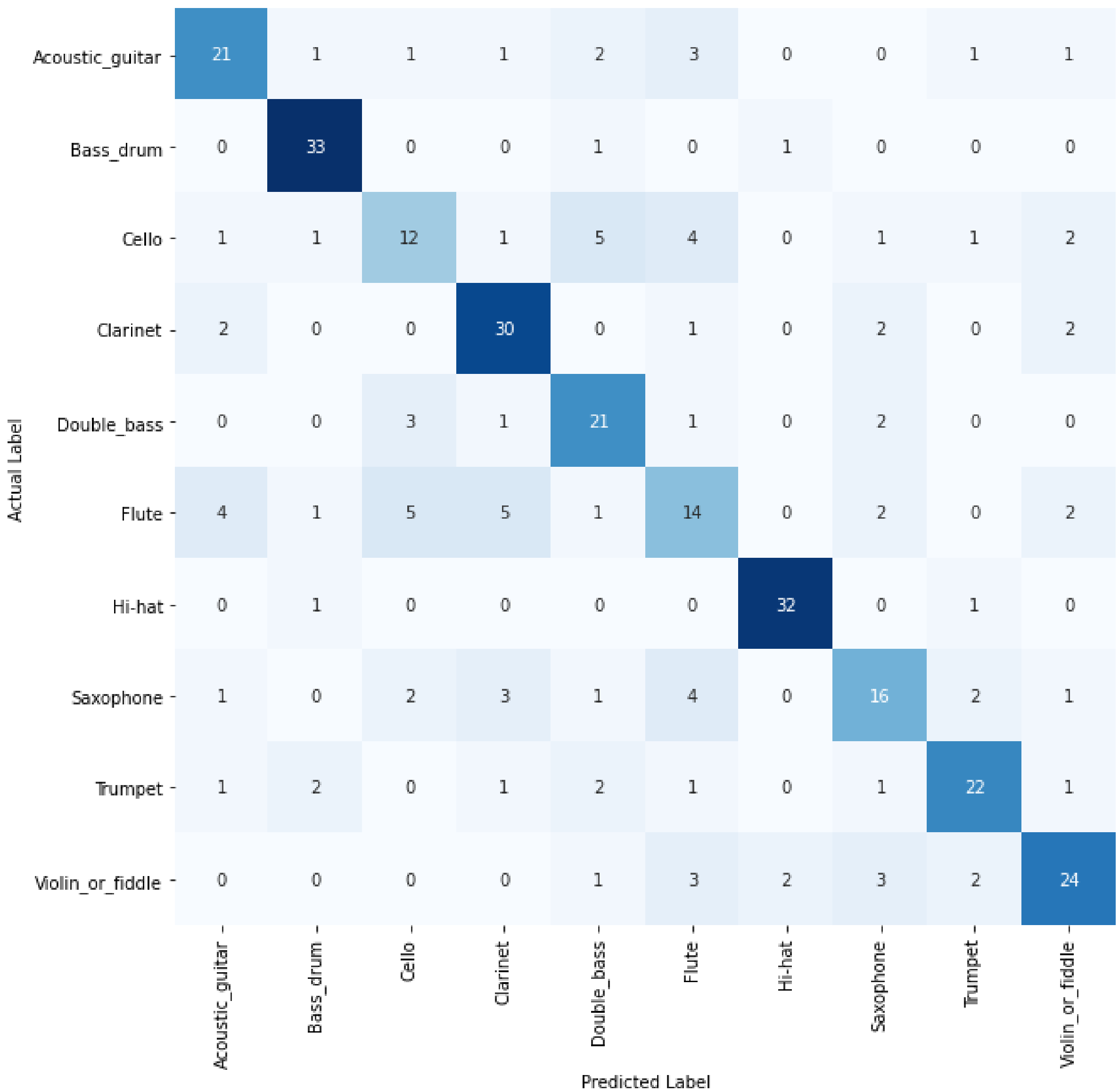
```
Epoch 1/30
44/44 [==============================] - 32s 722ms/step - loss: 8.9629 - acc:
0.1482 - val_loss: 2.0022 - val_acc: 0.3498
Epoch 2/30
44/44 [==============================] - 32s 719ms/step - loss: 2.1218 - acc:
0.2896 - val_loss: 1.8471 - val_acc: 0.3591
Epoch 3/30
44/44 [==============================] - 32s 718ms/step - loss: 1.9061 - acc:
0.3565 - val_loss: 1.8041 - val_acc: 0.4241
Epoch 4/30
44/44 [==============================] - 32s 723ms/step - loss: 1.8030 - acc:
0.4068 - val_loss: 1.6219 - val_acc: 0.4923
Epoch 5/30
44/44 [==============================] - 32s 722ms/step - loss: 1.7110 - acc:
0.4167 - val_loss: 1.4639 - val_acc: 0.5511
Epoch 6/30
44/44 [==============================] - 32s 721ms/step - loss: 1.6485 - acc:
0.4688 - val_loss: 1.4367 - val_acc: 0.5449
Epoch 7/30
44/44 [==============================] - 32s 719ms/step - loss: 1.4994 - acc:
0.4973 - val_loss: 1.3668 - val_acc: 0.5759
Epoch 8/30
44/44 [==============================] - 32s 720ms/step - loss: 1.3974 - acc:
0.5677 - val_loss: 1.2956 - val_acc: 0.5975
Epoch 9/30
44/44 [==============================] - 32s 725ms/step - loss: 1.3747 - acc:
0.5501 - val_loss: 1.2543 - val_acc: 0.6099
Epoch 10/30
44/44 [==============================] - 32s 721ms/step - loss: 1.2072 - acc:
0.5914 - val_loss: 1.1737 - val_acc: 0.6285
Epoch 11/30
44/44 [==============================] - 32s 718ms/step - loss: 1.1330 - acc:
0.6356 - val_loss: 1.2978 - val_acc: 0.5882
Epoch 12/30
44/44 [==============================] - 32s 718ms/step - loss: 1.1102 - acc:
0.6378 - val_loss: 1.2521 - val_acc: 0.6223
Epoch 13/30
44/44 [==============================] - 32s 727ms/step - loss: 1.0086 - acc:
0.6578 - val_loss: 1.1697 - val_acc: 0.6316
Epoch 14/30
44/44 [==============================] - 32s 717ms/step - loss: 1.0026 - acc:
0.6966 - val_loss: 1.1190 - val_acc: 0.6563
Epoch 15/30
44/44 [==============================] - 32s 719ms/step - loss: 0.8884 - acc:
0.7178 - val_loss: 1.0821 - val_acc: 0.6811
Epoch 16/30
44/44 [==============================] - 32s 717ms/step - loss: 0.8115 - acc:
0.7484 - val_loss: 1.1294 - val_acc: 0.6718
Epoch 17/30
44/44 [==============================] - 32s 720ms/step - loss: 0.6899 - acc:
0.7804 - val_loss: 1.1743 - val_acc: 0.6471
Epoch 18/30
44/44 [==============================] - 32s 724ms/step - loss: 0.7258 - acc:
0.7676 - val_loss: 1.1061 - val_acc: 0.6749
Epoch 19/30
44/44 [==============================] - 32s 724ms/step - loss: 0.7394 - acc:
0.7596 - val_loss: 1.0497 - val_acc: 0.6935
```

```
Epoch 20/30
44/44 [==============================] - 32s 720ms/step - loss: 0.5986 - acc:
0.8003 - val_loss: 1.1340 - val_acc: 0.7059
Epoch 21/30
44/44 [==============================] - 32s 720ms/step - loss: 0.6038 - acc:
0.8154 - val_loss: 1.2123 - val_acc: 0.6594
Epoch 22/30
44/44 [==============================] - 31s 713ms/step - loss: 0.5399 - acc:
0.8417 - val_loss: 1.1737 - val_acc: 0.6718
Epoch 23/30
44/44 [==============================] - 32s 717ms/step - loss: 0.5211 - acc:
0.8317 - val_loss: 1.2120 - val_acc: 0.6594
Epoch 24/30
44/44 [==============================] - 32s 721ms/step - loss: 0.4910 - acc:
0.8626 - val_loss: 1.1302 - val_acc: 0.6718
Epoch 25/30
44/44 [==============================] - 32s 724ms/step - loss: 0.5184 - acc:
0.8326 - val_loss: 1.1667 - val_acc: 0.6625
Epoch 26/30
44/44 [==============================] - 32s 719ms/step - loss: 0.4711 - acc:
0.8565 - val_loss: 1.3806 - val_acc: 0.6563
Epoch 27/30
44/44 [==============================] - 32s 717ms/step - loss: 0.4330 - acc:
0.8752 - val_loss: 1.4039 - val_acc: 0.6656
Epoch 28/30
44/44 [==============================] - 31s 716ms/step - loss: 0.4343 - acc:
0.8761 - val_loss: 1.2295 - val_acc: 0.6842
Epoch 29/30
44/44 [==============================] - 31s 716ms/step - loss: 0.4835 - acc:
0.8591 - val_loss: 1.2584 - val_acc: 0.6935
Epoch 30/30
44/44 [==============================] - 31s 714ms/step - loss: 0.5122 - acc:
0.8545 - val_loss: 1.1514 - val_acc: 0.6966
Loss: 0.4427550137042999
Validation Loss: 1.1514136791229248
```




```
Accuracy: 0.875812292098999
Validation Accuracy: 0.6965944170951843
```


Accuracy
Accuracy
Validation Accuracy
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0
5
10
15
20
25
30



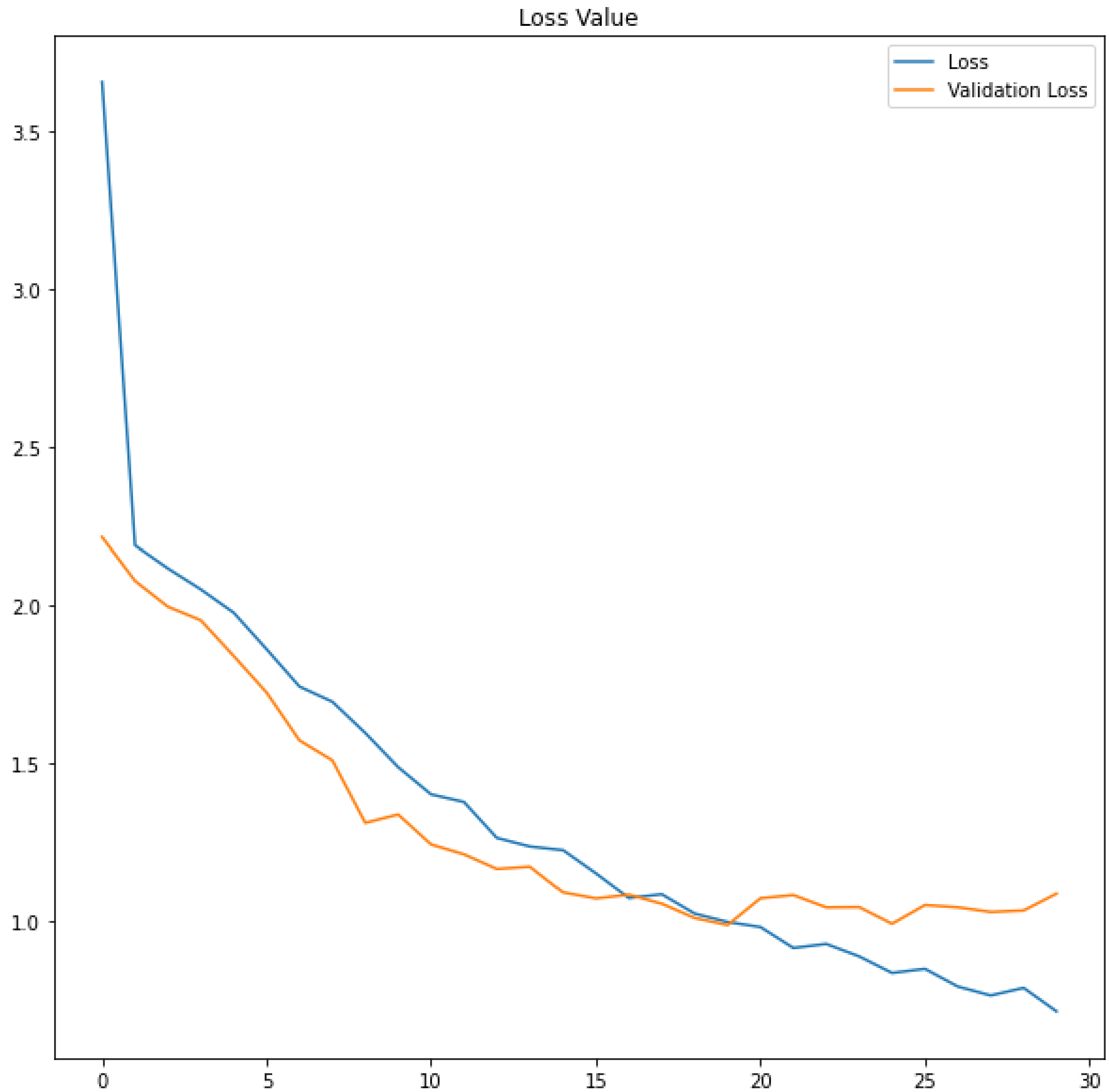
Actual Label
Predicted Label
Acoustic_guitar
Bass_drum
Cello
Clarinet
Double_bass
Flute
Hi-hat
Saxophone
Trumpet
Violin_or_fiddle
21 1 1 1 2 3 0 0 1 1
0 33 0 0 1 0 1 0 0 0
1 1 12 1 5 4 0 1 1 2
2 0 0 30 0 1 0 2 0 2
0 0 3 1 21 1 0 2 0 0
4 1 5 5 1 14 0 2 0 2
0 1 0 0 0 0 32 0 1 0
1 0 2 3 1 4 0 16 2 1
1 2 0 1 2 1 0 1 22 1
0 0 0 0 1 3 2 3 2 24

In [89]:
```
# Train/Validation/Test split
X_train, X_validation, y_train, Y_validation = mfcc_features_training_dataset,
mfcc_features_validation_dataset, one_hot_encoded_a, one_hot_encoded_b
X_test, y_test = mfcc_features_testing_dataset, one_hot_encoded_c

# Defining input shape for the neural network
input_shape = (X_train.shape[1], X_train.shape[2], 1)

# Reshape X_train and X_validation such that they are having the same shape as
the input shape
X_train = X_train.reshape(X_train.shape[0], X_train.shape[1], X_train.shape[2
], 1)
X_validation = X_validation.reshape(X_validation.shape[0], X_validation.shape[
1], X_validation.shape[2], 1)
X_test = X_test.reshape(X_test.shape[0], X_test.shape[1], X_test.shape[2], 1)

# Constructing the neural network architecture
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', strides=(1, 1),
    padding='same', input_shape=input_shape))
model.add(Conv2D(64, (3, 3), activation='relu', strides=(1, 1),
    padding='same'))
model.add(MaxPool2D((3, 3)))
model.add(Dropout(0.5))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(10, activation='softmax'))

model.compile(loss = 'categorical_crossentropy',
     optimizer = 'adam',
     metrics = ['acc'])

# Training the model
history = model.fit(X_train, y_train, epochs = 30, validation_data = (X_valida
tion, Y_validation))

# Displaying loss values
plt.figure(figsize = (10, 10))
plt.title('Loss Value')
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.legend(['Loss', 'Validation Loss'])
print('Loss:', history.history['loss'][-1])
print('Validation Loss:', history.history['val_loss'][-1])
plt.show()

# Displaying accuracy scores
plt.figure(figsize=(10, 10))
plt.title('Accuracy')
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.legend(['Accuracy', 'Validation Accuracy'])
print('Accuracy:', history.history['acc'][-1])
```

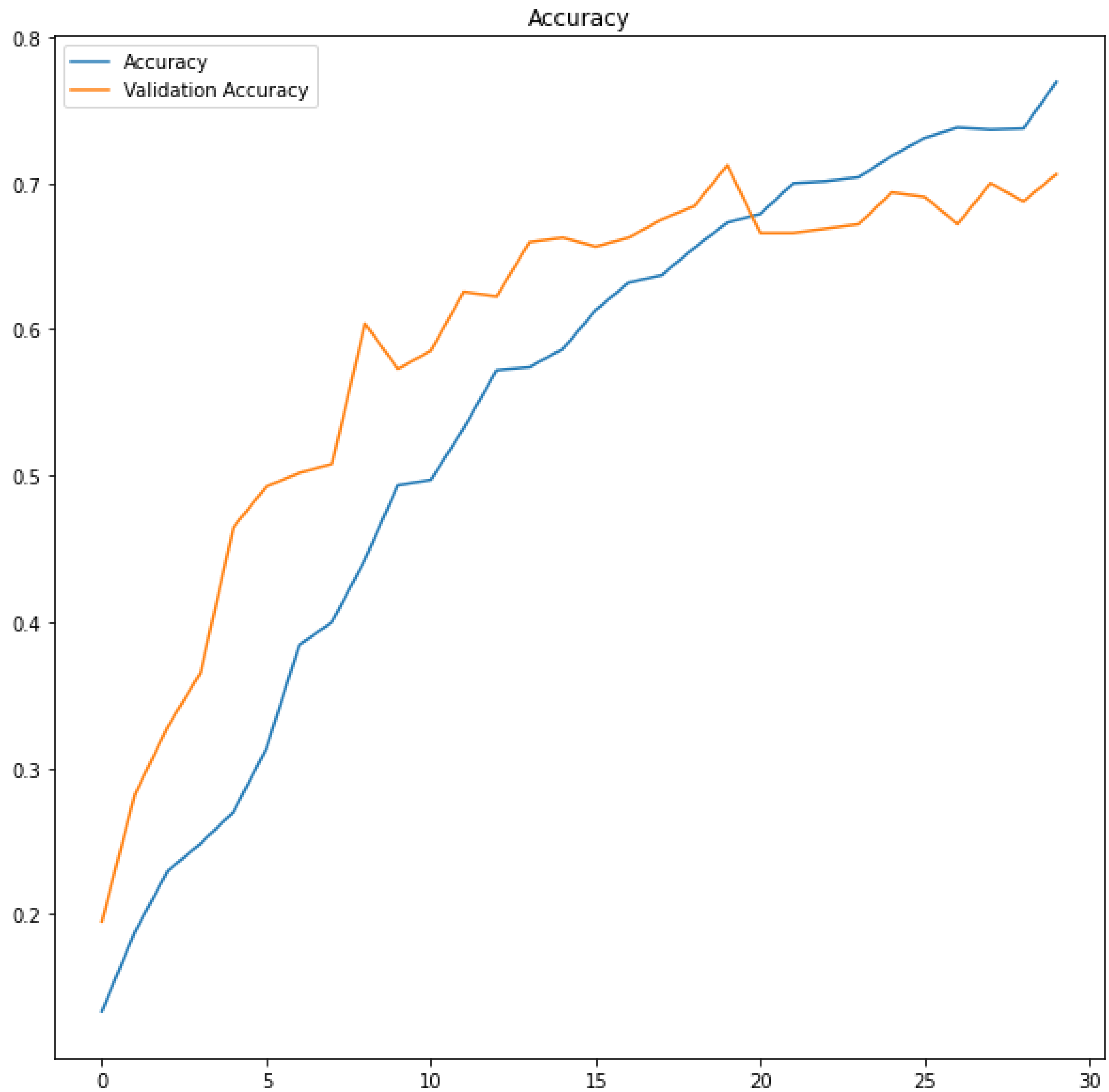
```
print('Validation Accuracy:', history.history['val_acc'][-1])
plt.show()

# Model evaluation
predictions = model.predict(X_validation)

predictions = np.argmax(predictions, axis=1)
Y_validation = one_hot_encoder_b.inverse_transform(Y_validation)

# Creating confusion matrix
cm = confusion_matrix(Y_validation, predictions)
plt.figure(figsize = (10, 10))
sns.heatmap(cm, annot = True, xticklabels = label_encoder_a.classes_, yticklab
els = label_encoder_b.classes_, fmt = 'd', cmap = plt.cm.Blues, cbar = False)
plt.xlabel('Predicted Label')
plt.ylabel('Actual Label')
plt.show()
```

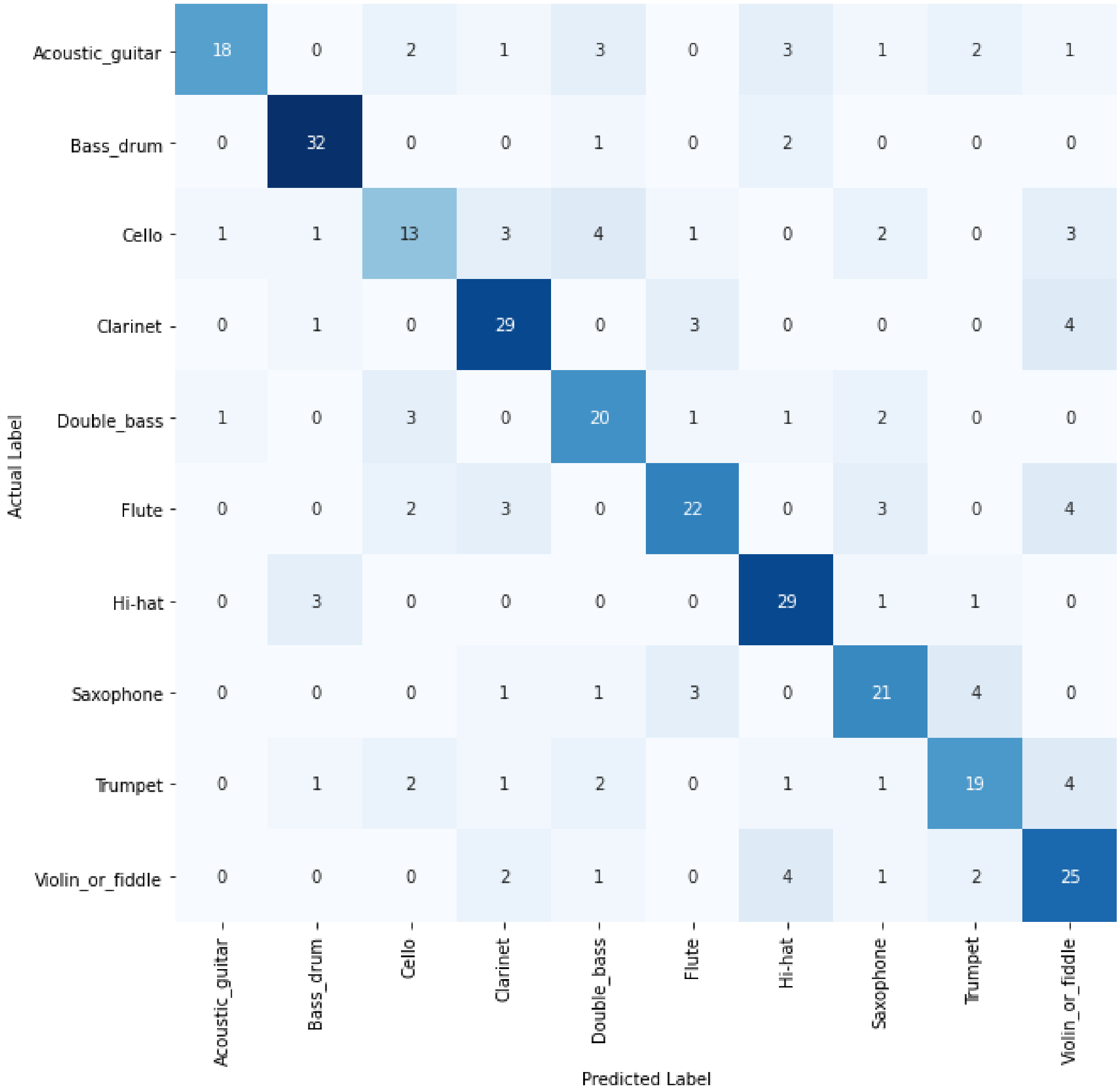
```
Epoch 1/30
44/44 [==============================] - 19s 420ms/step - loss: 6.3042 - acc:
0.1148 - val_loss: 2.2129 - val_acc: 0.1950
Epoch 2/30
44/44 [==============================] - 18s 414ms/step - loss: 2.2149 - acc:
0.1814 - val_loss: 2.0747 - val_acc: 0.2817
Epoch 3/30
44/44 [==============================] - 18s 417ms/step - loss: 2.1321 - acc:
0.2143 - val_loss: 1.9930 - val_acc: 0.3282
Epoch 4/30
44/44 [==============================] - 18s 415ms/step - loss: 2.0572 - acc:
0.2434 - val_loss: 1.9502 - val_acc: 0.3653
Epoch 5/30
44/44 [==============================] - 18s 417ms/step - loss: 2.0001 - acc:
0.2709 - val_loss: 1.8370 - val_acc: 0.4644
Epoch 6/30
44/44 [==============================] - 18s 418ms/step - loss: 1.8475 - acc:
0.3202 - val_loss: 1.7212 - val_acc: 0.4923
Epoch 7/30
44/44 [==============================] - 18s 416ms/step - loss: 1.7398 - acc:
0.3966 - val_loss: 1.5694 - val_acc: 0.5015
Epoch 8/30
44/44 [==============================] - 18s 417ms/step - loss: 1.7394 - acc:
0.3718 - val_loss: 1.5063 - val_acc: 0.5077
Epoch 9/30
44/44 [==============================] - 19s 432ms/step - loss: 1.5833 - acc:
0.4426 - val_loss: 1.3082 - val_acc: 0.6037
Epoch 10/30
44/44 [==============================] - 18s 418ms/step - loss: 1.4707 - acc:
0.4886 - val_loss: 1.3347 - val_acc: 0.5728
Epoch 11/30
44/44 [==============================] - 18s 417ms/step - loss: 1.4118 - acc:
0.4845 - val_loss: 1.2398 - val_acc: 0.5851
Epoch 12/30
44/44 [==============================] - 18s 416ms/step - loss: 1.3761 - acc:
0.5327 - val_loss: 1.2085 - val_acc: 0.6254
Epoch 13/30
44/44 [==============================] - 18s 417ms/step - loss: 1.3101 - acc:
0.5527 - val_loss: 1.1620 - val_acc: 0.6223
Epoch 14/30
44/44 [==============================] - 18s 417ms/step - loss: 1.1752 - acc:
0.5861 - val_loss: 1.1691 - val_acc: 0.6594
Epoch 15/30
44/44 [==============================] - 18s 417ms/step - loss: 1.2312 - acc:
0.5824 - val_loss: 1.0879 - val_acc: 0.6625
Epoch 16/30
44/44 [==============================] - 18s 418ms/step - loss: 1.0798 - acc:
0.6395 - val_loss: 1.0689 - val_acc: 0.6563
Epoch 17/30
44/44 [==============================] - 18s 417ms/step - loss: 1.0793 - acc:
0.6195 - val_loss: 1.0805 - val_acc: 0.6625
Epoch 18/30
44/44 [==============================] - 18s 416ms/step - loss: 1.1234 - acc:
0.6185 - val_loss: 1.0520 - val_acc: 0.6749
Epoch 19/30
44/44 [==============================] - 18s 417ms/step - loss: 1.0201 - acc:
0.6500 - val_loss: 1.0068 - val_acc: 0.6842
```

```
Epoch 20/30
44/44 [==============================] - 18s 414ms/step - loss: 1.0014 - acc:
0.6812 - val_loss: 0.9843 - val_acc: 0.7121
Epoch 21/30
44/44 [==============================] - 18s 417ms/step - loss: 0.9896 - acc:
0.6761 - val_loss: 1.0693 - val_acc: 0.6656
Epoch 22/30
44/44 [==============================] - 18s 415ms/step - loss: 0.9484 - acc:
0.6900 - val_loss: 1.0794 - val_acc: 0.6656
Epoch 23/30
44/44 [==============================] - 18s 418ms/step - loss: 0.9420 - acc:
0.6846 - val_loss: 1.0405 - val_acc: 0.6687
Epoch 24/30
44/44 [==============================] - 18s 415ms/step - loss: 0.8343 - acc:
0.7133 - val_loss: 1.0412 - val_acc: 0.6718
Epoch 25/30
44/44 [==============================] - 18s 417ms/step - loss: 0.8646 - acc:
0.7031 - val_loss: 0.9884 - val_acc: 0.6935
Epoch 26/30
44/44 [==============================] - 18s 418ms/step - loss: 0.8497 - acc:
0.7261 - val_loss: 1.0477 - val_acc: 0.6904
Epoch 27/30
44/44 [==============================] - 18s 417ms/step - loss: 0.7750 - acc:
0.7390 - val_loss: 1.0406 - val_acc: 0.6718
Epoch 28/30
44/44 [==============================] - 18s 418ms/step - loss: 0.7443 - acc:
0.7382 - val_loss: 1.0258 - val_acc: 0.6997
Epoch 29/30
44/44 [==============================] - 18s 419ms/step - loss: 0.7682 - acc:
0.7443 - val_loss: 1.0305 - val_acc: 0.6873
Epoch 30/30
44/44 [==============================] - 18s 419ms/step - loss: 0.7100 - acc:
0.7565 - val_loss: 1.0838 - val_acc: 0.7059
Loss: 0.7108391523361206
Validation Loss: 1.0838122367858887
```




```
Accuracy: 0.7689530849456787
Validation Accuracy: 0.7058823704719543
```


Accuracy
Accuracy
Validation Accuracy
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0
5
10
15
20
25
30


Acoustic_guitar
Bass_drum
Cello
Clarinet
Double_bass
Flute
Hi-hat
Saxophone
Trumpet
Violin_or_fiddle
Actual Label
Predicted Label

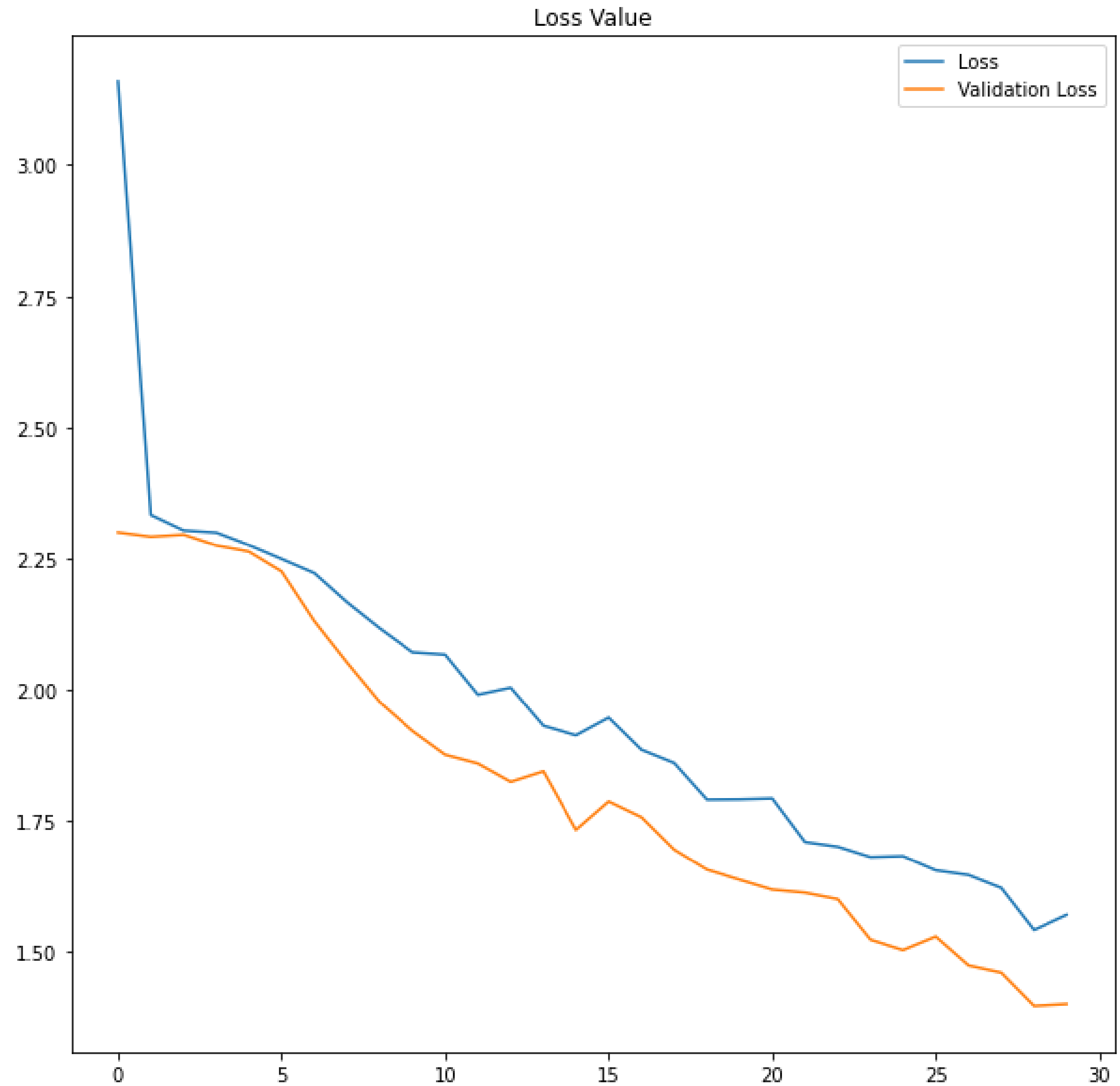
In [90]:

```
# Train/Validation/Test split
X_train, X_validation, y_train, Y_validation = mfcc_features_training_dataset,
mfcc_features_validation_dataset, one_hot_encoded_a, one_hot_encoded_b
X_test, y_test = mfcc_features_testing_dataset, one_hot_encoded_c

# Defining input shape for the neural network
input_shape = (X_train.shape[1], X_train.shape[2], 1)

# Reshape X_train and X_validation such that they are having the same shape as
the input shape
X_train = X_train.reshape(X_train.shape[0], X_train.shape[1], X_train.shape[2
], 1)
X_validation = X_validation.reshape(X_validation.shape[0], X_validation.shape[
1], X_validation.shape[2], 1)
X_test = X_test.reshape(X_test.shape[0], X_test.shape[1], X_test.shape[2], 1)

# Constructing the neural network architecture
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', strides=(2, 2),
    padding='same', input_shape=input_shape))
model.add(Conv2D(64, (3, 3), activation='relu', strides=(2, 2),
    padding='same'))
model.add(MaxPool2D((3, 3)))
model.add(Dropout(0.5))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(32, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(10, activation='softmax'))

model.compile(loss = 'categorical_crossentropy',
     optimizer = 'adam',
     metrics = ['acc'])

# Training the model
history = model.fit(X_train, y_train, epochs = 30, validation_data = (X_valida
tion, Y_validation))

# Displaying loss values
plt.figure(figsize = (10, 10))
plt.title('Loss Value')
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.legend(['Loss', 'Validation Loss'])
print('Loss:', history.history['loss'][-1])
print('Validation Loss:', history.history['val_loss'][-1])
plt.show()

# Displaying accuracy scores
plt.figure(figsize=(10, 10))
plt.title('Accuracy')
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
```

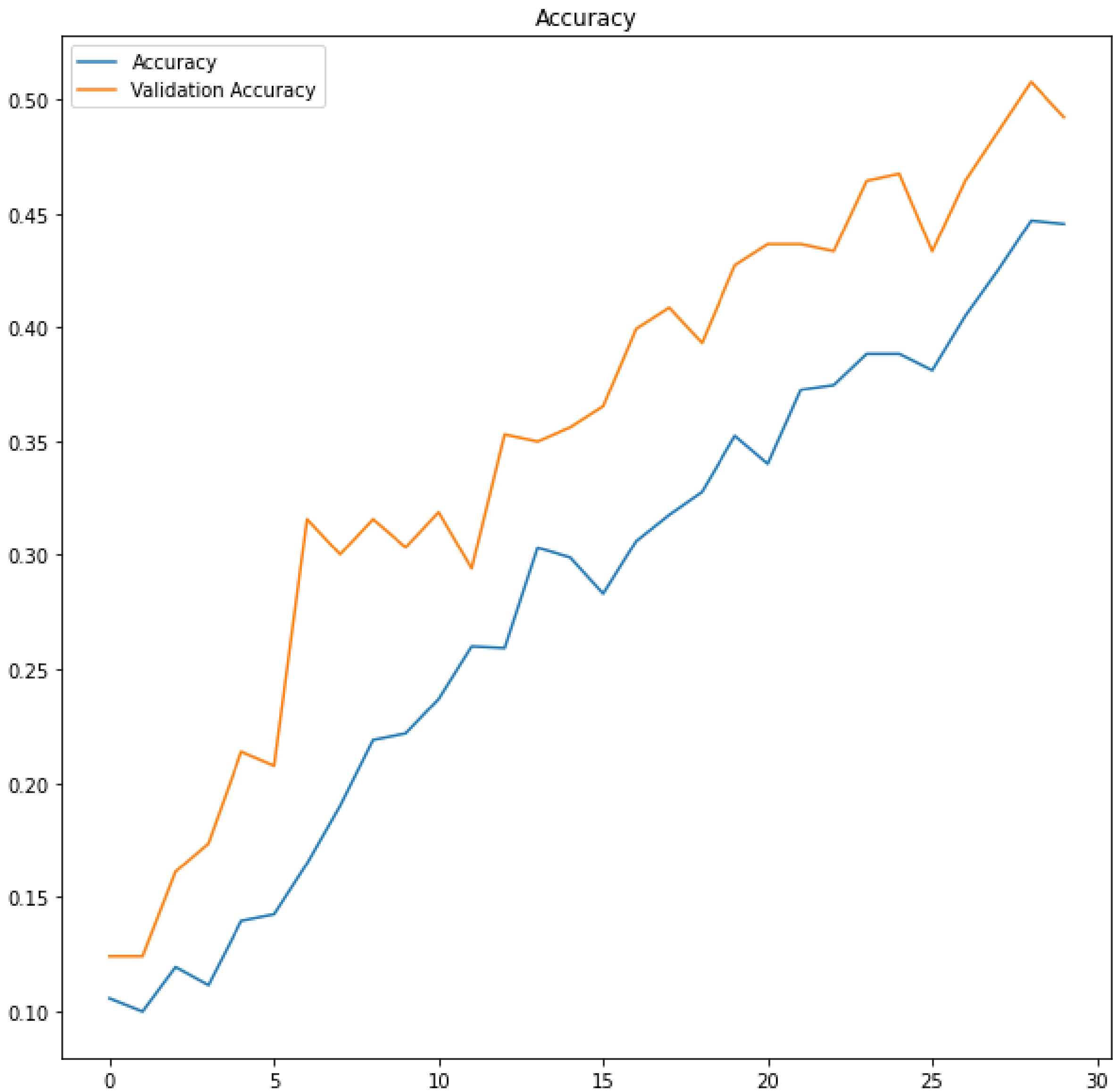
```
plt.legend(['Accuracy', 'Validation Accuracy'])
print('Accuracy:', history.history['acc'][-1])
print('Validation Accuracy:', history.history['val_acc'][-1])
plt.show()

# Model evaluation
predictions = model.predict(X_validation)

predictions = np.argmax(predictions, axis=1)
Y_validation = one_hot_encoder_b.inverse_transform(Y_validation)

# Creating confusion matrix
cm = confusion_matrix(Y_validation, predictions)
plt.figure(figsize = (10, 10))
sns.heatmap(cm, annot = True, xticklabels = label_encoder_a.classes_, yticklab
els = label_encoder_b.classes_, fmt = 'd', cmap = plt.cm.Blues, cbar = False)
plt.xlabel('Predicted Label')
plt.ylabel('Actual Label')
plt.show()
```

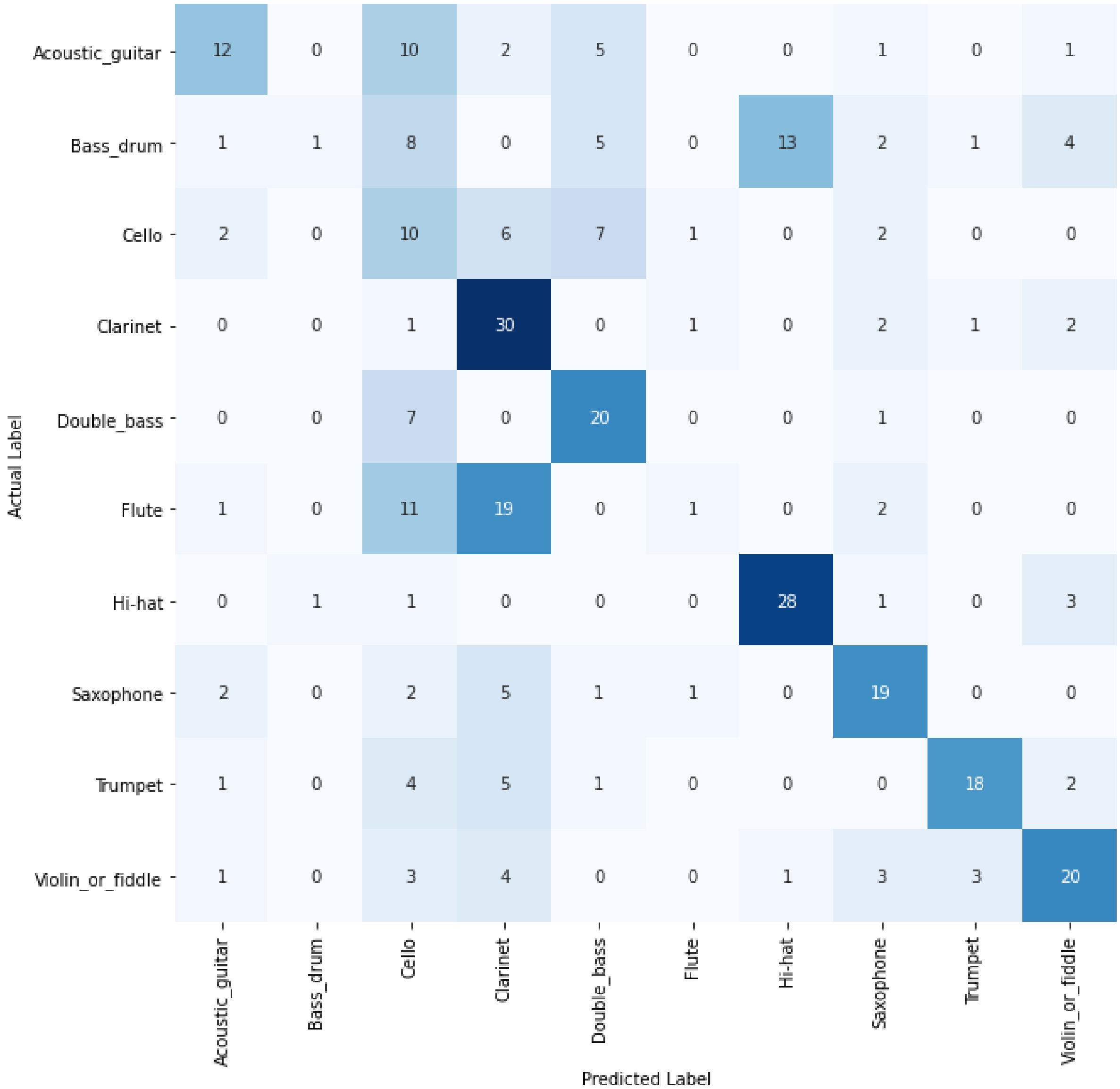
```
Epoch 1/30
44/44 [==============================] - 4s 54ms/step - loss: 4.4349 - acc:
0.1021 - val_loss: 2.2990 - val_acc: 0.1238
Epoch 2/30
44/44 [==============================] - 2s 48ms/step - loss: 2.3315 - acc:
0.0919 - val_loss: 2.2908 - val_acc: 0.1238
Epoch 3/30
44/44 [==============================] - 2s 48ms/step - loss: 2.3105 - acc:
0.1277 - val_loss: 2.2947 - val_acc: 0.1610
Epoch 4/30
44/44 [==============================] - 2s 49ms/step - loss: 2.2933 - acc:
0.1144 - val_loss: 2.2739 - val_acc: 0.1734
Epoch 5/30
44/44 [==============================] - 2s 49ms/step - loss: 2.2718 - acc:
0.1460 - val_loss: 2.2622 - val_acc: 0.2136
Epoch 6/30
44/44 [==============================] - 2s 48ms/step - loss: 2.2493 - acc:
0.1510 - val_loss: 2.2244 - val_acc: 0.2074
Epoch 7/30
44/44 [==============================] - 2s 48ms/step - loss: 2.2276 - acc:
0.1538 - val_loss: 2.1297 - val_acc: 0.3158
Epoch 8/30
44/44 [==============================] - 2s 48ms/step - loss: 2.1903 - acc:
0.1793 - val_loss: 2.0504 - val_acc: 0.3003
Epoch 9/30
44/44 [==============================] - 2s 48ms/step - loss: 2.1271 - acc:
0.2198 - val_loss: 1.9749 - val_acc: 0.3158
Epoch 10/30
44/44 [==============================] - 2s 49ms/step - loss: 2.0686 - acc:
0.2156 - val_loss: 1.9202 - val_acc: 0.3034
Epoch 11/30
44/44 [==============================] - 2s 48ms/step - loss: 2.0668 - acc:
0.2433 - val_loss: 1.8748 - val_acc: 0.3189
Epoch 12/30
44/44 [==============================] - 2s 48ms/step - loss: 1.9776 - acc:
0.2553 - val_loss: 1.8580 - val_acc: 0.2941
Epoch 13/30
44/44 [==============================] - 2s 49ms/step - loss: 2.0315 - acc:
0.2519 - val_loss: 1.8232 - val_acc: 0.3529
Epoch 14/30
44/44 [==============================] - 2s 48ms/step - loss: 1.9260 - acc:
0.3087 - val_loss: 1.8432 - val_acc: 0.3498
Epoch 15/30
44/44 [==============================] - 2s 49ms/step - loss: 1.9278 - acc:
0.2866 - val_loss: 1.7316 - val_acc: 0.3560
Epoch 16/30
44/44 [==============================] - 2s 49ms/step - loss: 1.9248 - acc:
0.2856 - val_loss: 1.7857 - val_acc: 0.3653
Epoch 17/30
44/44 [==============================] - 2s 48ms/step - loss: 1.8872 - acc:
0.3018 - val_loss: 1.7557 - val_acc: 0.3994
Epoch 18/30
44/44 [==============================] - 2s 48ms/step - loss: 1.8825 - acc:
0.3198 - val_loss: 1.6937 - val_acc: 0.4087
Epoch 19/30
44/44 [==============================] - 2s 49ms/step - loss: 1.8058 - acc:
0.3162 - val_loss: 1.6558 - val_acc: 0.3932
```

```
Epoch 20/30
44/44 [==============================] - 2s 47ms/step - loss: 1.7907 - acc:
0.3527 - val_loss: 1.6367 - val_acc: 0.4272
Epoch 21/30
44/44 [==============================] - 2s 48ms/step - loss: 1.7890 - acc:
0.3324 - val_loss: 1.6176 - val_acc: 0.4365
Epoch 22/30
44/44 [==============================] - 2s 49ms/step - loss: 1.7117 - acc:
0.3781 - val_loss: 1.6118 - val_acc: 0.4365
Epoch 23/30
44/44 [==============================] - 2s 49ms/step - loss: 1.6779 - acc:
0.3877 - val_loss: 1.5993 - val_acc: 0.4334
Epoch 24/30
44/44 [==============================] - 2s 48ms/step - loss: 1.6827 - acc:
0.3793 - val_loss: 1.5219 - val_acc: 0.4644
Epoch 25/30
44/44 [==============================] - 2s 48ms/step - loss: 1.6492 - acc:
0.4058 - val_loss: 1.5022 - val_acc: 0.4675
Epoch 26/30
44/44 [==============================] - 2s 48ms/step - loss: 1.6863 - acc:
0.3604 - val_loss: 1.5280 - val_acc: 0.4334
Epoch 27/30
44/44 [==============================] - 2s 49ms/step - loss: 1.5810 - acc:
0.4294 - val_loss: 1.4731 - val_acc: 0.4644
Epoch 28/30
44/44 [==============================] - 2s 47ms/step - loss: 1.6136 - acc:
0.4330 - val_loss: 1.4593 - val_acc: 0.4861
Epoch 29/30
44/44 [==============================] - 2s 48ms/step - loss: 1.5489 - acc:
0.4544 - val_loss: 1.3960 - val_acc: 0.5077
Epoch 30/30
44/44 [==============================] - 2s 48ms/step - loss: 1.5518 - acc:
0.4372 - val_loss: 1.3997 - val_acc: 0.4923
Loss: 1.5693082809448242
Validation Loss: 1.3996778726577759
```




```
Accuracy: 0.4454873502254486
Validation Accuracy: 0.49226006865501404
```


Accuracy
Accuracy
Validation Accuracy
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0
5
10
15
20
25
30


Actual Label
Acoustic_guitar 12 0 10 2 5 0 0 1 0 1
Bass_drum 1 1 8 0 5 0 13 2 1 4
Cello 2 0 10 6 7 1 0 2 0 0
Clarinet 0 0 1 30 0 1 0 2 1 2
Double_bass 0 0 7 0 20 0 0 1 0 0
Flute 1 0 11 19 0 1 0 2 0 0
Hi-hat 0 1 1 0 0 0 28 1 0 3
Saxophone 2 0 2 5 1 1 0 19 0 0
Trumpet 1 0 4 5 1 0 0 0 18 2
Violin_or_fiddle 1 0 3 4 0 0 1 3 3 20
Acoustic_guitar
Bass_drum
Cello
Clarinet
Double_bass
Flute
Hi-hat
Saxophone
Trumpet
Violin_or_fiddle
Predicted Label

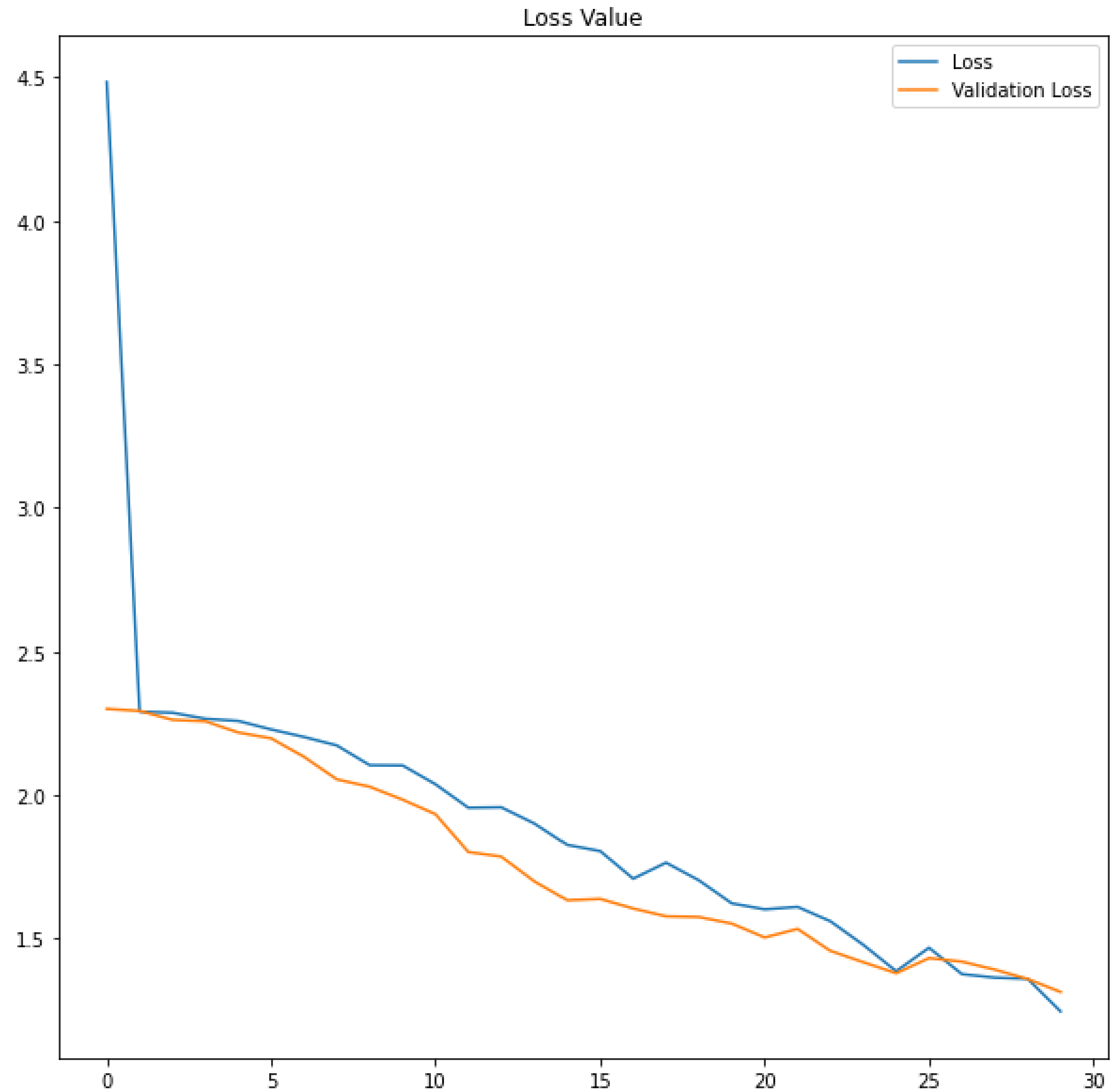
In [91]:

```
# Train/Validation/Test split
X_train, X_validation, y_train, Y_validation = mfcc_features_training_dataset,
mfcc_features_validation_dataset, one_hot_encoded_a, one_hot_encoded_b
X_test, y_test = mfcc_features_testing_dataset, one_hot_encoded_c

# Defining input shape for the neural network
input_shape = (X_train.shape[1], X_train.shape[2], 1)

# Reshape X_train and X_validation such that they are having the same shape as
the input shape
X_train = X_train.reshape(X_train.shape[0], X_train.shape[1], X_train.shape[2
], 1)
X_validation = X_validation.reshape(X_validation.shape[0], X_validation.shape[
1], X_validation.shape[2], 1)
X_test = X_test.reshape(X_test.shape[0], X_test.shape[1], X_test.shape[2], 1)

# Constructing the neural network architecture
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', strides=(1, 1),
    padding='same', input_shape=input_shape))
model.add(Conv2D(64, (3, 3), activation='relu', strides=(1, 1),
    padding='same'))
model.add(MaxPool2D((3, 3)))
model.add(Dropout(0.5))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(32, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(10, activation='softmax'))

model.compile(loss = 'categorical_crossentropy',
     optimizer = 'adam',
     metrics = ['acc'])

# Training the model
history = model.fit(X_train, y_train, epochs = 30, validation_data = (X_valida
tion, Y_validation))

# Displaying loss values
plt.figure(figsize = (10, 10))
plt.title('Loss Value')
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.legend(['Loss', 'Validation Loss'])
print('Loss:', history.history['loss'][-1])
print('Validation Loss:', history.history['val_loss'][-1])
plt.show()

# Displaying accuracy scores
plt.figure(figsize=(10, 10))
plt.title('Accuracy')
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
```

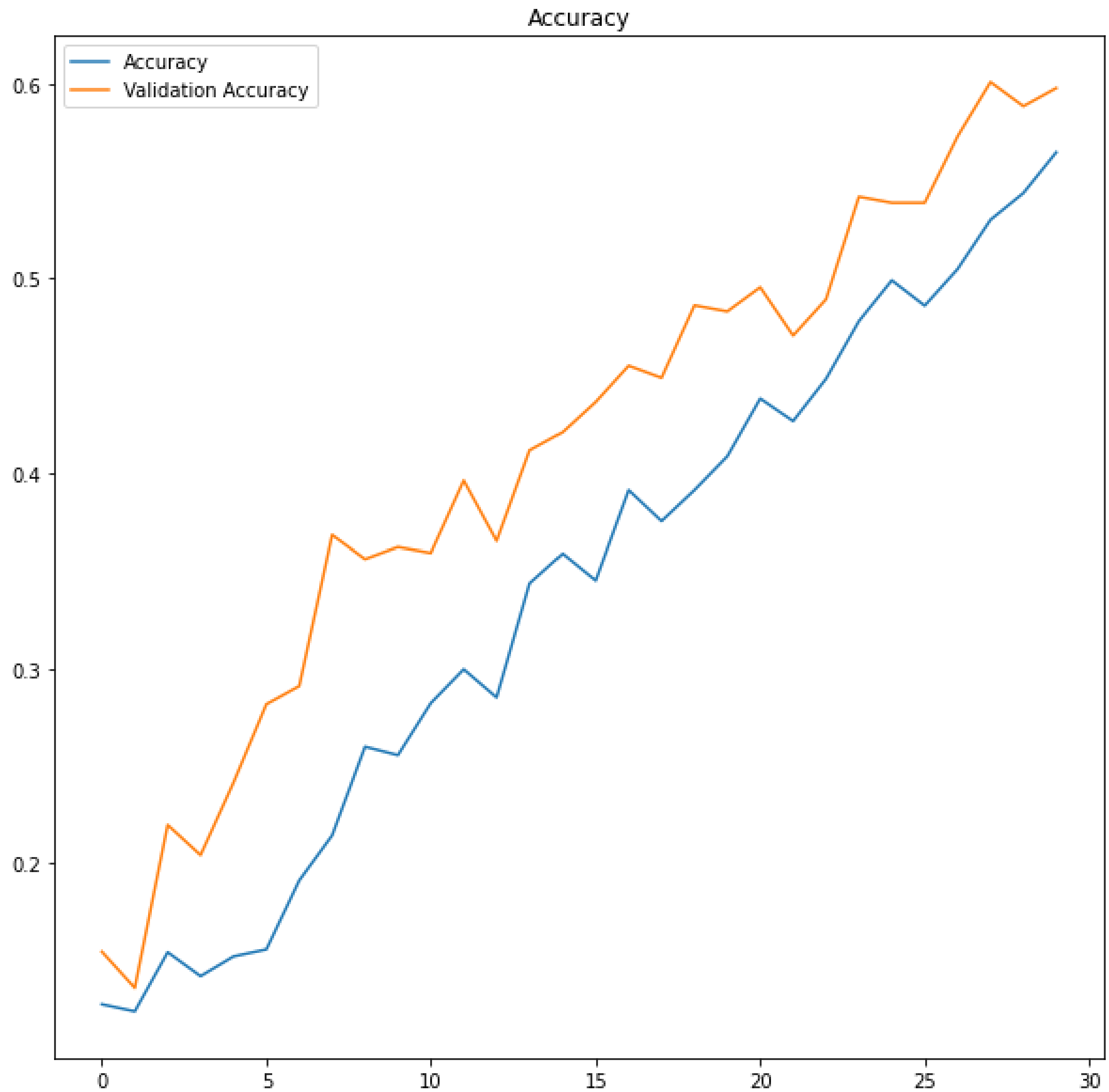
```
plt.legend(['Accuracy', 'Validation Accuracy'])
print('Accuracy:', history.history['acc'][-1])
print('Validation Accuracy:', history.history['val_acc'][-1])
plt.show()

# Model evaluation
predictions = model.predict(X_validation)

predictions = np.argmax(predictions, axis=1)
Y_validation = one_hot_encoder_b.inverse_transform(Y_validation)

# Creating confusion matrix
cm = confusion_matrix(Y_validation, predictions)
plt.figure(figsize = (10, 10))
sns.heatmap(cm, annot = True, xticklabels = label_encoder_a.classes_, yticklab
els = label_encoder_b.classes_, fmt = 'd', cmap = plt.cm.Blues, cbar = False)
plt.xlabel('Predicted Label')
plt.ylabel('Actual Label')
plt.show()
```

```
Epoch 1/30
44/44 [==============================] - 19s 420ms/step - loss: 8.1865 - acc:
0.1231 - val_loss: 2.2990 - val_acc: 0.1548
Epoch 2/30
44/44 [==============================] - 18s 416ms/step - loss: 2.2867 - acc:
0.1294 - val_loss: 2.2921 - val_acc: 0.1362
Epoch 3/30
44/44 [==============================] - 18s 415ms/step - loss: 2.2846 - acc:
0.1510 - val_loss: 2.2610 - val_acc: 0.2198
Epoch 4/30
44/44 [==============================] - 18s 417ms/step - loss: 2.2614 - acc:
0.1366 - val_loss: 2.2558 - val_acc: 0.2043
Epoch 5/30
44/44 [==============================] - 18s 416ms/step - loss: 2.2642 - acc:
0.1617 - val_loss: 2.2168 - val_acc: 0.2415
Epoch 6/30
44/44 [==============================] - 18s 416ms/step - loss: 2.2282 - acc:
0.1497 - val_loss: 2.1964 - val_acc: 0.2817
Epoch 7/30
44/44 [==============================] - 18s 416ms/step - loss: 2.2263 - acc:
0.1833 - val_loss: 2.1321 - val_acc: 0.2910
Epoch 8/30
44/44 [==============================] - 18s 418ms/step - loss: 2.1918 - acc:
0.2052 - val_loss: 2.0533 - val_acc: 0.3684
Epoch 9/30
44/44 [==============================] - 18s 418ms/step - loss: 2.1065 - acc:
0.2419 - val_loss: 2.0277 - val_acc: 0.3560
Epoch 10/30
44/44 [==============================] - 18s 417ms/step - loss: 2.0995 - acc:
0.2613 - val_loss: 1.9826 - val_acc: 0.3622
Epoch 11/30
44/44 [==============================] - 18s 415ms/step - loss: 2.0284 - acc:
0.2931 - val_loss: 1.9320 - val_acc: 0.3591
Epoch 12/30
44/44 [==============================] - 18s 416ms/step - loss: 1.9580 - acc:
0.2995 - val_loss: 1.8001 - val_acc: 0.3963
Epoch 13/30
44/44 [==============================] - 18s 415ms/step - loss: 1.9727 - acc:
0.2845 - val_loss: 1.7842 - val_acc: 0.3653
Epoch 14/30
44/44 [==============================] - 18s 417ms/step - loss: 1.9134 - acc:
0.3331 - val_loss: 1.6984 - val_acc: 0.4118
Epoch 15/30
44/44 [==============================] - 19s 428ms/step - loss: 1.8465 - acc:
0.3495 - val_loss: 1.6320 - val_acc: 0.4211
Epoch 16/30
44/44 [==============================] - 18s 420ms/step - loss: 1.8259 - acc:
0.3285 - val_loss: 1.6366 - val_acc: 0.4365
Epoch 17/30
44/44 [==============================] - 18s 416ms/step - loss: 1.7177 - acc:
0.3867 - val_loss: 1.6032 - val_acc: 0.4551
Epoch 18/30
44/44 [==============================] - 18s 417ms/step - loss: 1.6958 - acc:
0.3878 - val_loss: 1.5760 - val_acc: 0.4489
Epoch 19/30
44/44 [==============================] - 18s 417ms/step - loss: 1.7084 - acc:
0.4010 - val_loss: 1.5735 - val_acc: 0.4861
```

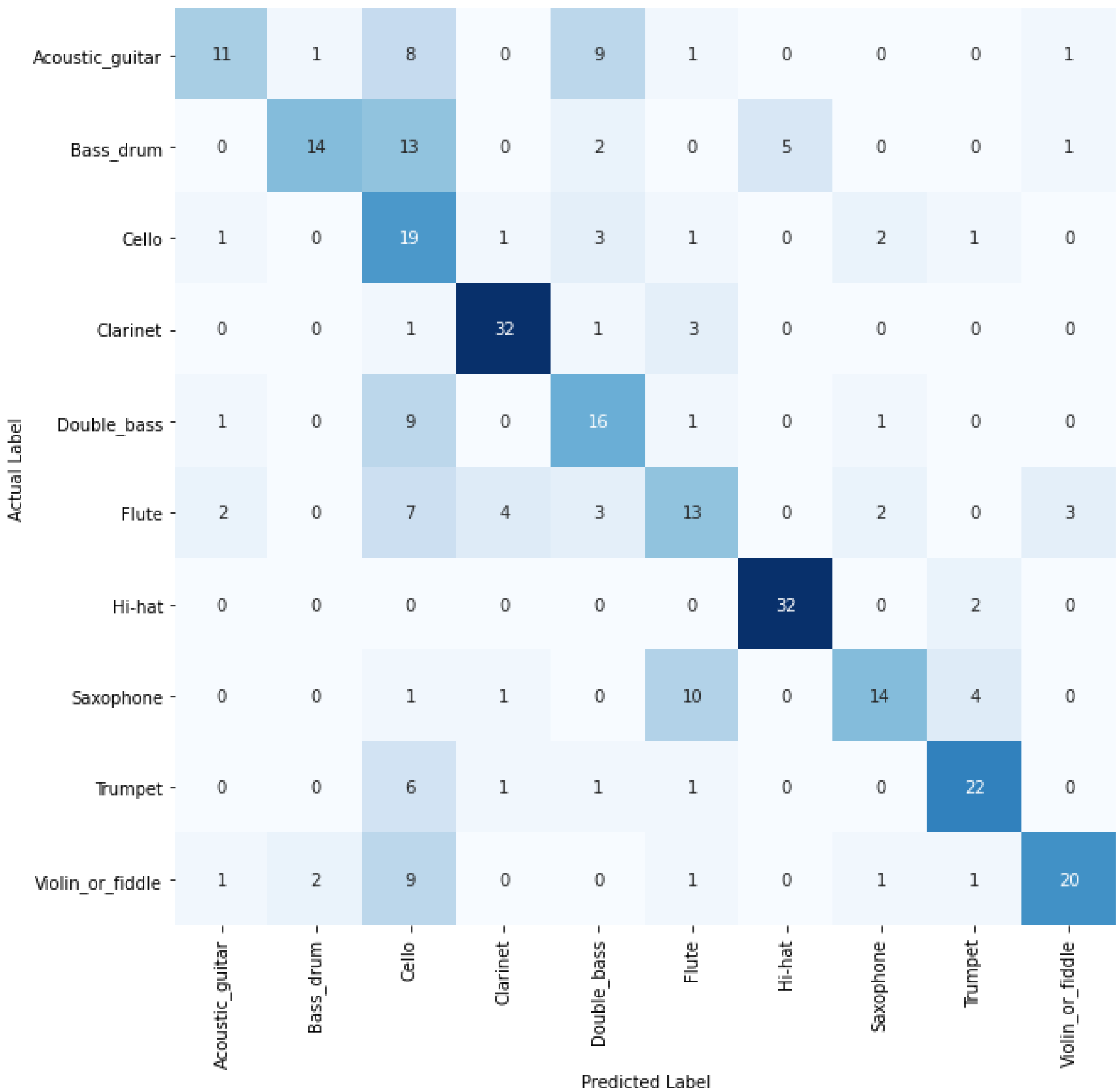
```
Epoch 20/30
44/44 [==============================] - 18s 417ms/step - loss: 1.6212 - acc: 0.4160 - val_loss: 1.5506 - val_acc: 0.4830
Epoch 21/30
44/44 [==============================] - 18s 418ms/step - loss: 1.5825 - acc: 0.4379 - val_loss: 1.5023 - val_acc: 0.4954
Epoch 22/30
44/44 [==============================] - 18s 416ms/step - loss: 1.6215 - acc: 0.4316 - val_loss: 1.5320 - val_acc: 0.4706
Epoch 23/30
44/44 [==============================] - 18s 415ms/step - loss: 1.5826 - acc: 0.4487 - val_loss: 1.4554 - val_acc: 0.4892
Epoch 24/30
44/44 [==============================] - 18s 416ms/step - loss: 1.4487 - acc: 0.4784 - val_loss: 1.4160 - val_acc: 0.5418
Epoch 25/30
44/44 [==============================] - 18s 418ms/step - loss: 1.4255 - acc: 0.4753 - val_loss: 1.3777 - val_acc: 0.5387
Epoch 26/30
44/44 [==============================] - 18s 418ms/step - loss: 1.3596 - acc: 0.5225 - val_loss: 1.4302 - val_acc: 0.5387
Epoch 27/30
44/44 [==============================] - 18s 418ms/step - loss: 1.3874 - acc: 0.5158 - val_loss: 1.4180 - val_acc: 0.5728
Epoch 28/30
44/44 [==============================] - 18s 418ms/step - loss: 1.3458 - acc: 0.5170 - val_loss: 1.3901 - val_acc: 0.6006
Epoch 29/30
44/44 [==============================] - 18s 419ms/step - loss: 1.3502 - acc: 0.5292 - val_loss: 1.3574 - val_acc: 0.5882
Epoch 30/30
44/44 [==============================] - 18s 418ms/step - loss: 1.2594 - acc: 0.5407 - val_loss: 1.3122 - val_acc: 0.5975
Loss: 1.2446188926696777
Validation Loss: 1.3122127056121826
```




```
Accuracy: 0.5646209120750427
Validation Accuracy: 0.5975232124328613
```



Accuracy
Accuracy
Validation Accuracy
0.6
0.5
0.4
0.3
0.2
0
5
10
15
20
25
30



Actual Label
Predicted Label
Acoustic_guitar
Bass_drum
Cello
Clarinet
Double_bass
Flute
Hi-hat
Saxophone
Trumpet
Violin_or_fiddle
11 1 8 0 9 1 0 0 0 1
0 14 13 0 2 0 5 0 0 1
1 0 19 1 3 1 0 2 1 0
0 0 1 32 1 3 0 0 0 0
1 0 9 0 16 1 0 1 0 0
2 0 7 4 3 13 0 2 0 3
0 0 0 0 0 0 32 0 2 0
0 0 1 1 0 10 0 14 4 0
0 0 6 1 1 1 0 0 22 0
1 2 9 0 0 1 0 1 1 20

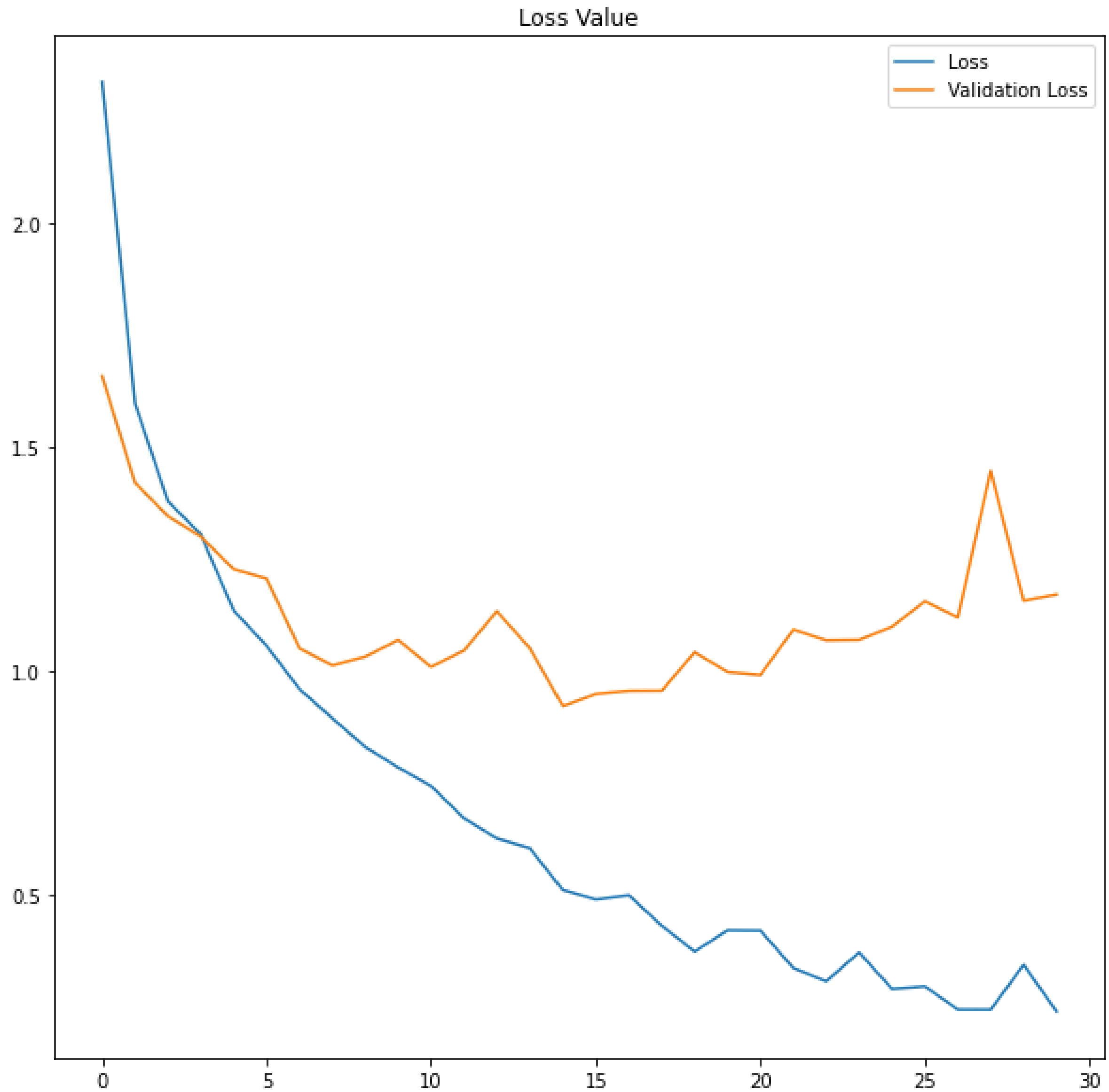
In [92]:

```
# Train/Validation/Test split
X_train, X_validation, y_train, Y_validation = mfcc_features_training_dataset,
mfcc_features_validation_dataset, one_hot_encoded_a, one_hot_encoded_b
X_test, y_test = mfcc_features_testing_dataset, one_hot_encoded_c

# Defining input shape for the neural network
input_shape = (X_train.shape[1], X_train.shape[2], 1)

# Reshape X_train and X_validation such that they are having the same shape as
the input shape
X_train = X_train.reshape(X_train.shape[0], X_train.shape[1], X_train.shape[2
], 1)
X_validation = X_validation.reshape(X_validation.shape[0], X_validation.shape[
1], X_validation.shape[2], 1)
X_test = X_test.reshape(X_test.shape[0], X_test.shape[1], X_test.shape[2], 1)

# Constructing the neural network architecture
model = Sequential()
model.add(Conv2D(32, (4, 4), activation='relu', strides=(2, 2),
    padding='same', input_shape=input_shape))
model.add(MaxPool2D((2, 2)))
model.add(Conv2D(64, (4, 4), activation='relu', strides=(2, 2),
    padding='same'))
model.add(MaxPool2D((2, 2)))
model.add(Dropout(0.5))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(10, activation='softmax'))

model.compile(loss = 'categorical_crossentropy',
     optimizer = 'adam',
     metrics = ['acc'])

# Training the model
history = model.fit(X_train, y_train, epochs = 30, validation_data = (X_valida
tion, Y_validation))

# Displaying loss values
plt.figure(figsize = (10, 10))
plt.title('Loss Value')
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.legend(['Loss', 'Validation Loss'])
print('Loss:', history.history['loss'][-1])
print('Validation Loss:', history.history['val_loss'][-1])
plt.show()

# Displaying accuracy scores
plt.figure(figsize=(10, 10))
plt.title('Accuracy')
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.legend(['Accuracy', 'Validation Accuracy'])
print('Accuracy:', history.history['acc'][-1])
print('Validation Accuracy:', history.history['val_acc'][-1])
```

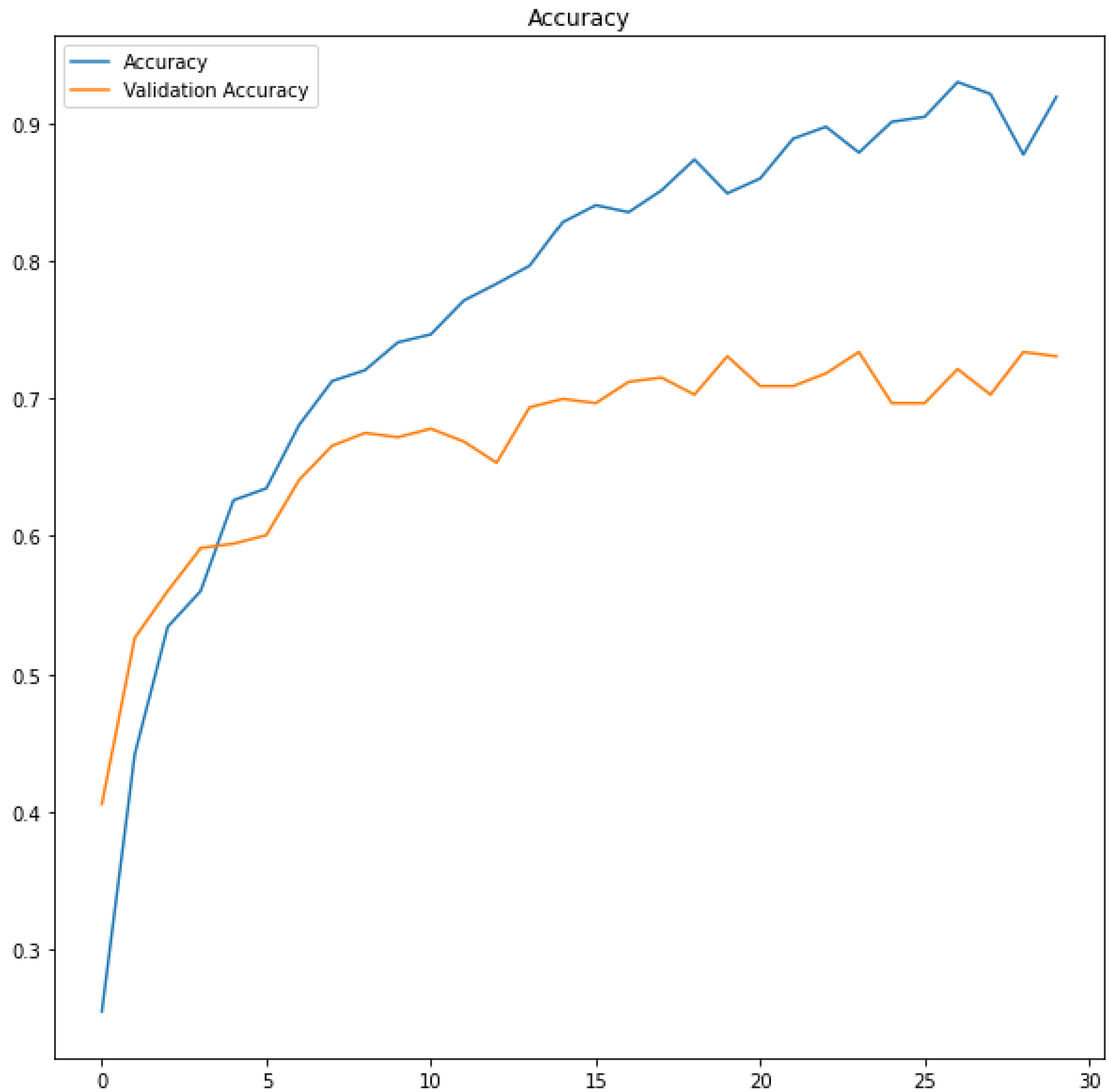
```
plt.show()

# Model evaluation
predictions = model.predict(X_validation)

predictions = np.argmax(predictions, axis=1)
Y_validation = one_hot_encoder_b.inverse_transform(Y_validation)

# Creating confusion matrix
cm = confusion_matrix(Y_validation, predictions)
plt.figure(figsize = (10, 10))
sns.heatmap(cm, annot = True, xticklabels = label_encoder_a.classes_, yticklab
els = label_encoder_b.classes_, fmt = 'd', cmap = plt.cm.Blues, cbar = False)
plt.xlabel('Predicted Label')
plt.ylabel('Actual Label')
plt.show()
```

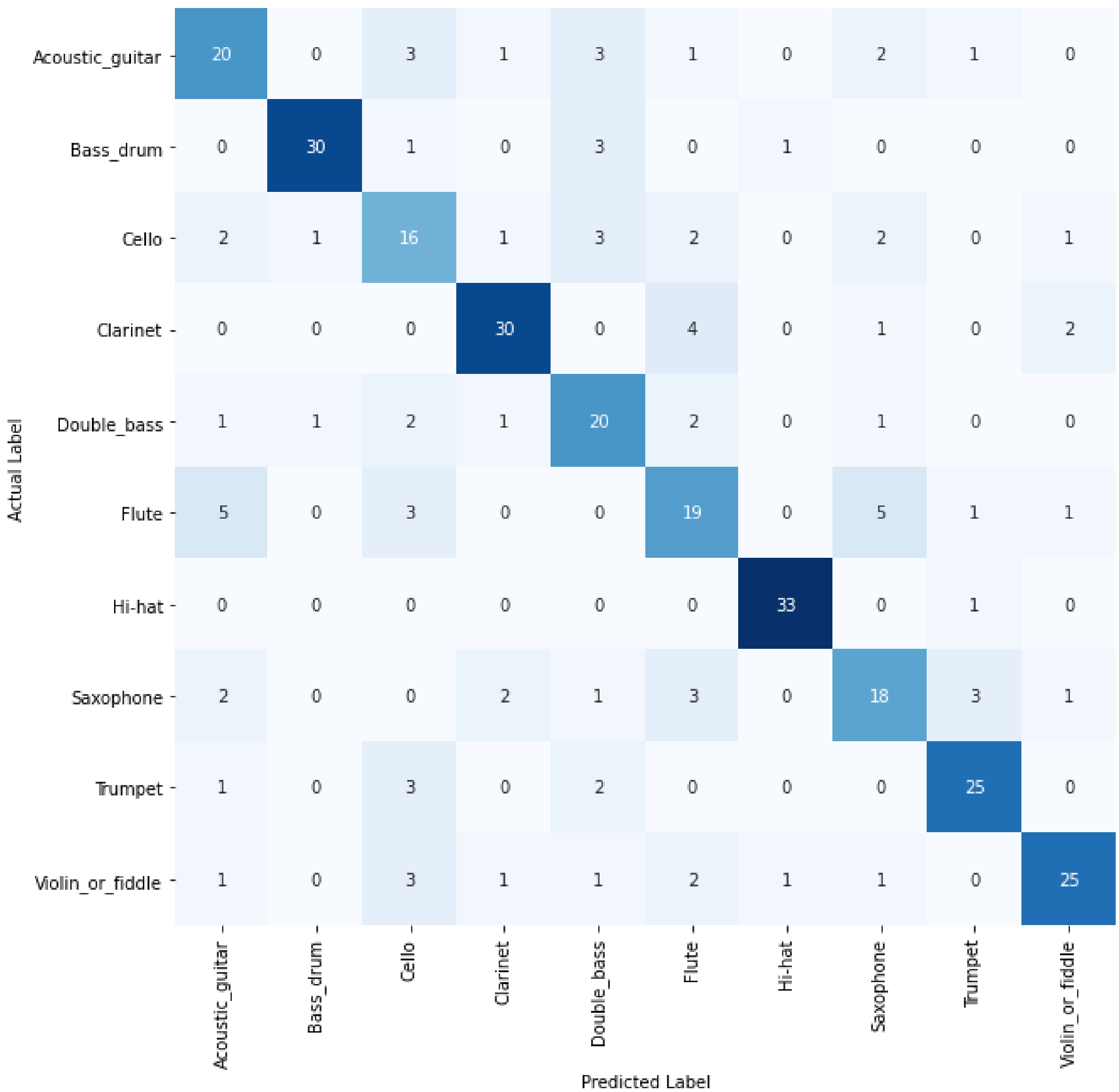
```
Epoch 1/30
44/44 [==============================] - 2s 43ms/step - loss: 2.9567 - acc:
0.1983 - val_loss: 1.6587 - val_acc: 0.4056
Epoch 2/30
44/44 [==============================] - 2s 39ms/step - loss: 1.6511 - acc:
0.4157 - val_loss: 1.4204 - val_acc: 0.5263
Epoch 3/30
44/44 [==============================] - 2s 39ms/step - loss: 1.4723 - acc:
0.4893 - val_loss: 1.3457 - val_acc: 0.5604
Epoch 4/30
44/44 [==============================] - 2s 39ms/step - loss: 1.3137 - acc:
0.5599 - val_loss: 1.2999 - val_acc: 0.5913
Epoch 5/30
44/44 [==============================] - 2s 39ms/step - loss: 1.1322 - acc:
0.6255 - val_loss: 1.2281 - val_acc: 0.5944
Epoch 6/30
44/44 [==============================] - 2s 38ms/step - loss: 1.0385 - acc:
0.6376 - val_loss: 1.2070 - val_acc: 0.6006
Epoch 7/30
44/44 [==============================] - 2s 39ms/step - loss: 0.9722 - acc:
0.6729 - val_loss: 1.0510 - val_acc: 0.6409
Epoch 8/30
44/44 [==============================] - 2s 39ms/step - loss: 0.8855 - acc:
0.7084 - val_loss: 1.0128 - val_acc: 0.6656
Epoch 9/30
44/44 [==============================] - 2s 38ms/step - loss: 0.8036 - acc:
0.7251 - val_loss: 1.0323 - val_acc: 0.6749
Epoch 10/30
44/44 [==============================] - 2s 39ms/step - loss: 0.7567 - acc:
0.7502 - val_loss: 1.0697 - val_acc: 0.6718
Epoch 11/30
44/44 [==============================] - 2s 39ms/step - loss: 0.7142 - acc:
0.7599 - val_loss: 1.0095 - val_acc: 0.6780
Epoch 12/30
44/44 [==============================] - 2s 40ms/step - loss: 0.6644 - acc:
0.7677 - val_loss: 1.0461 - val_acc: 0.6687
Epoch 13/30
44/44 [==============================] - 2s 40ms/step - loss: 0.6667 - acc:
0.7554 - val_loss: 1.1338 - val_acc: 0.6533
Epoch 14/30
44/44 [==============================] - 2s 38ms/step - loss: 0.6090 - acc:
0.8072 - val_loss: 1.0516 - val_acc: 0.6935
Epoch 15/30
44/44 [==============================] - 2s 39ms/step - loss: 0.5342 - acc:
0.8205 - val_loss: 0.9219 - val_acc: 0.6997
Epoch 16/30
44/44 [==============================] - 2s 40ms/step - loss: 0.5036 - acc:
0.8401 - val_loss: 0.9490 - val_acc: 0.6966
Epoch 17/30
44/44 [==============================] - 2s 39ms/step - loss: 0.4822 - acc:
0.8381 - val_loss: 0.9559 - val_acc: 0.7121
Epoch 18/30
44/44 [==============================] - 2s 39ms/step - loss: 0.4141 - acc:
0.8545 - val_loss: 0.9563 - val_acc: 0.7152
Epoch 19/30
44/44 [==============================] - 2s 39ms/step - loss: 0.3449 - acc:
0.8872 - val_loss: 1.0422 - val_acc: 0.7028
```

```
Epoch 20/30
44/44 [==============================] - 2s 39ms/step - loss: 0.4506 - acc:
0.8477 - val_loss: 0.9981 - val_acc: 0.7307
Epoch 21/30
44/44 [==============================] - 2s 39ms/step - loss: 0.4660 - acc:
0.8466 - val_loss: 0.9914 - val_acc: 0.7090
Epoch 22/30
44/44 [==============================] - 2s 40ms/step - loss: 0.3339 - acc:
0.8884 - val_loss: 1.0930 - val_acc: 0.7090
Epoch 23/30
44/44 [==============================] - 2s 39ms/step - loss: 0.2992 - acc:
0.9092 - val_loss: 1.0689 - val_acc: 0.7183
Epoch 24/30
44/44 [==============================] - 2s 39ms/step - loss: 0.3378 - acc:
0.8890 - val_loss: 1.0699 - val_acc: 0.7337
Epoch 25/30
44/44 [==============================] - 2s 39ms/step - loss: 0.2682 - acc:
0.9088 - val_loss: 1.0993 - val_acc: 0.6966
Epoch 26/30
44/44 [==============================] - 2s 39ms/step - loss: 0.2855 - acc:
0.9058 - val_loss: 1.1562 - val_acc: 0.6966
Epoch 27/30
44/44 [==============================] - 2s 40ms/step - loss: 0.2518 - acc:
0.9250 - val_loss: 1.1201 - val_acc: 0.7214
Epoch 28/30
44/44 [==============================] - 2s 39ms/step - loss: 0.2473 - acc:
0.9204 - val_loss: 1.4466 - val_acc: 0.7028
Epoch 29/30
44/44 [==============================] - 2s 38ms/step - loss: 0.3607 - acc:
0.8705 - val_loss: 1.1576 - val_acc: 0.7337
Epoch 30/30
44/44 [==============================] - 2s 39ms/step - loss: 0.2569 - acc:
0.9139 - val_loss: 1.1715 - val_acc: 0.7307
Loss: 0.23867852985858917
Validation Loss: 1.1714613437652588
```




```
Accuracy: 0.9191336035728455
Validation Accuracy: 0.7306501269340515
```


Accuracy
Accuracy
Validation Accuracy
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0
5
10
15
20
25
30

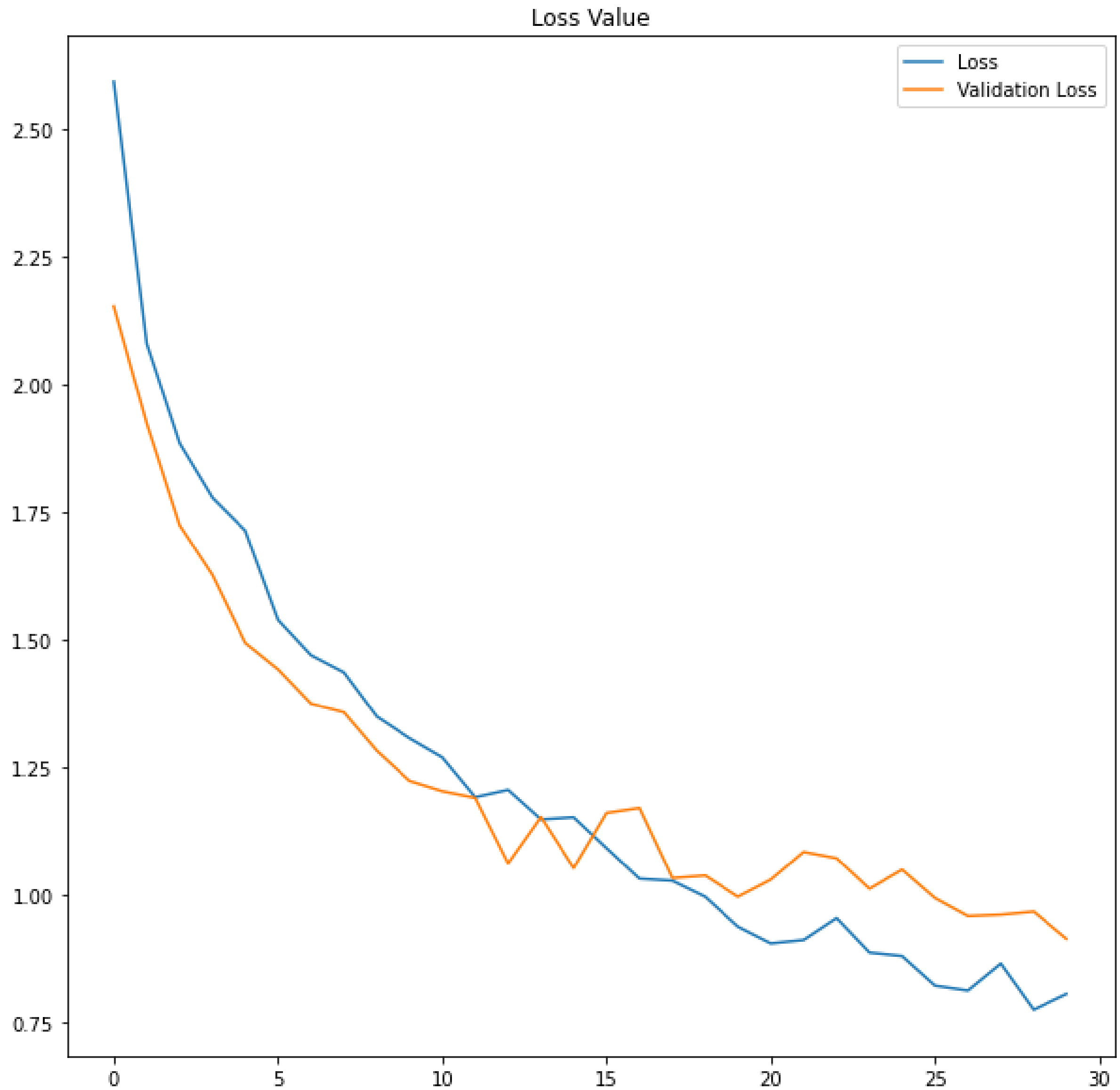

Acoustic_guitar
Bass_drum
Cello
Clarinet
Double_bass
Flute
Hi-hat
Saxophone
Trumpet
Violin_or_fiddle
20 0 3 1 3 1 0 2 1 0
0 30 1 0 3 0 1 0 0 0
2 1 16 1 3 2 0 2 0 1
0 0 0 30 0 4 0 1 0 2
1 1 2 1 20 2 0 1 0 0
5 0 3 0 0 19 0 5 1 1
0 0 0 0 0 0 33 0 1 0
2 0 0 2 1 3 0 18 3 1
1 0 3 0 2 0 0 0 25 0
1 0 3 1 1 2 1 1 0 25
Actual Label
Predicted Label

```
In [93]: # Train/Validation/Test split
X_train, X_validation, y_train, Y_validation = mfcc_features_training_dataset,
mfcc_features_validation_dataset, one_hot_encoded_a, one_hot_encoded_b
X_test, y_test = mfcc_features_testing_dataset, one_hot_encoded_c

# Defining input shape for the neural network
input_shape = (X_train.shape[1], X_train.shape[2], 1)

# Reshape X_train and X_validation such that they are having the same shape as
the input shape
X_train = X_train.reshape(X_train.shape[0], X_train.shape[1], X_train.shape[2
], 1)
X_validation = X_validation.reshape(X_validation.shape[0], X_validation.shape[
1], X_validation.shape[2], 1)
X_test = X_test.reshape(X_test.shape[0], X_test.shape[1], X_test.shape[2], 1)

# Constructing the neural network architecture
model = Sequential()
model.add(Conv2D(32, (4, 4), activation='relu', strides=(2, 2),
    padding='same', input_shape=input_shape))
model.add(MaxPool2D((2, 2)))
model.add(Conv2D(64, (4, 4), activation='relu', strides=(2, 2),
    padding='same'))
model.add(MaxPool2D((2, 2)))
model.add(Dropout(0.5))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(64, activation='relu'))
model.add(Dense(10, activation='softmax'))

model.compile(loss = 'categorical_crossentropy',
     optimizer = 'adam',
     metrics = ['acc'])

# Training the model
history = model.fit(X_train, y_train, epochs = 30, validation_data = (X_valida
tion, Y_validation))

# Displaying loss values
plt.figure(figsize = (10, 10))
plt.title('Loss Value')
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.legend(['Loss', 'Validation Loss'])
print('Loss:', history.history['loss'][-1])
print('Validation Loss:', history.history['val_loss'][-1])
plt.show()

# Displaying accuracy scores
plt.figure(figsize=(10, 10))
plt.title('Accuracy')
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.legend(['Accuracy', 'Validation Accuracy'])
print('Accuracy:', history.history['acc'][-1])
```

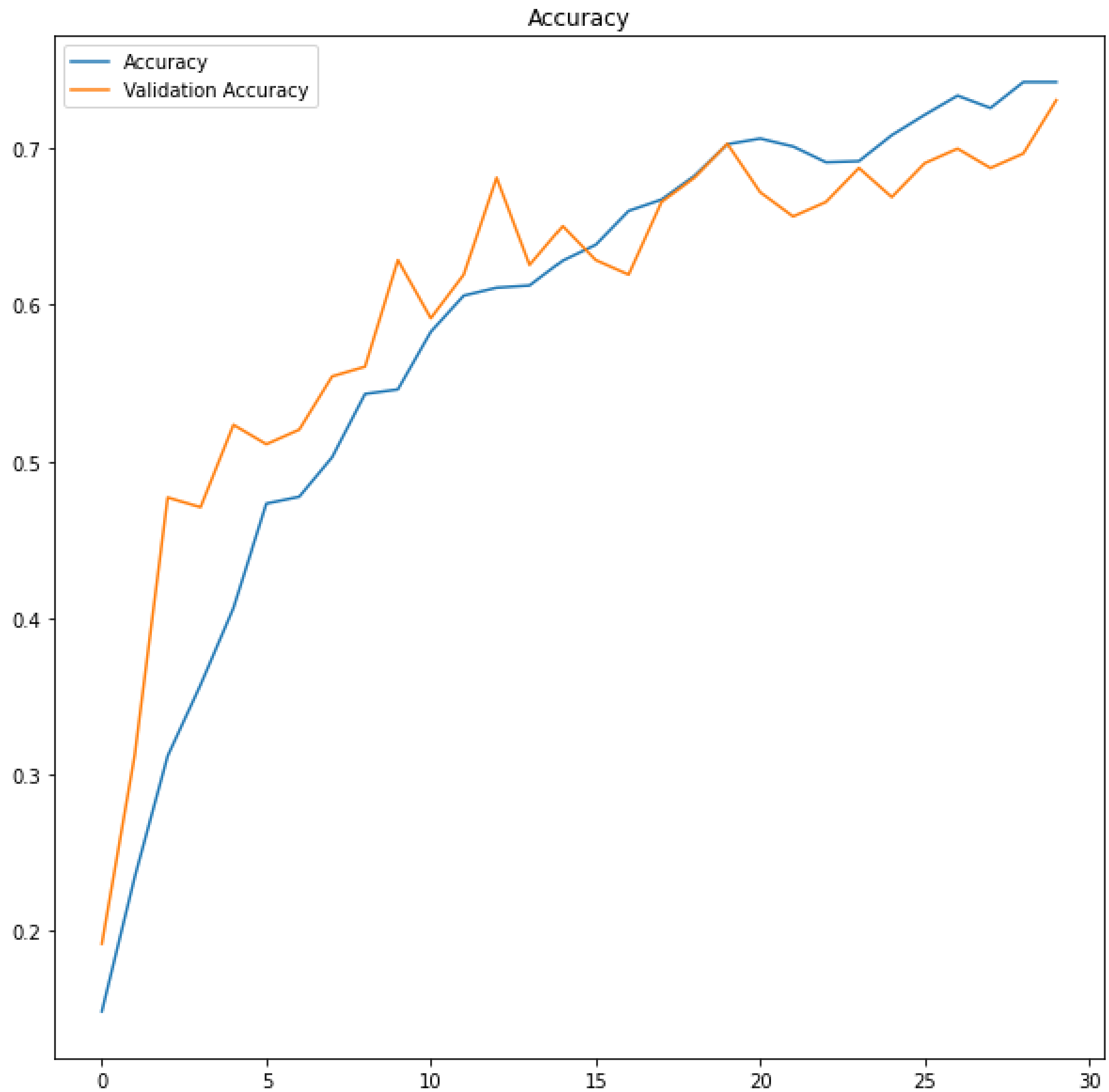
```
print('Validation Accuracy:', history.history['val_acc'][-1])
plt.show()

# Model evaluation
predictions = model.predict(X_validation)

predictions = np.argmax(predictions, axis=1)
Y_validation = one_hot_encoder_b.inverse_transform(Y_validation)

# Creating confusion matrix
cm = confusion_matrix(Y_validation, predictions)
plt.figure(figsize = (10, 10))
sns.heatmap(cm, annot = True, xticklabels = label_encoder_a.classes_, yticklab
els = label_encoder_b.classes_, fmt = 'd', cmap = plt.cm.Blues, cbar = False)
plt.xlabel('Predicted Label')
plt.ylabel('Actual Label')
plt.show()
```

```
Epoch 1/30
44/44 [==============================] - 3s 44ms/step - loss: 3.2481 - acc:
0.1393 - val_loss: 2.1537 - val_acc: 0.1920
Epoch 2/30
44/44 [==============================] - 2s 39ms/step - loss: 2.1056 - acc:
0.2348 - val_loss: 1.9250 - val_acc: 0.3127
Epoch 3/30
44/44 [==============================] - 2s 38ms/step - loss: 1.8944 - acc:
0.3025 - val_loss: 1.7251 - val_acc: 0.4768
Epoch 4/30
44/44 [==============================] - 2s 39ms/step - loss: 1.7754 - acc:
0.3632 - val_loss: 1.6275 - val_acc: 0.4706
Epoch 5/30
44/44 [==============================] - 2s 39ms/step - loss: 1.7687 - acc:
0.3871 - val_loss: 1.4949 - val_acc: 0.5232
Epoch 6/30
44/44 [==============================] - 2s 39ms/step - loss: 1.5351 - acc:
0.4712 - val_loss: 1.4429 - val_acc: 0.5108
Epoch 7/30
44/44 [==============================] - 2s 39ms/step - loss: 1.4887 - acc:
0.4716 - val_loss: 1.3758 - val_acc: 0.5201
Epoch 8/30
44/44 [==============================] - 2s 40ms/step - loss: 1.4080 - acc:
0.5119 - val_loss: 1.3589 - val_acc: 0.5542
Epoch 9/30
44/44 [==============================] - 2s 39ms/step - loss: 1.3371 - acc:
0.5396 - val_loss: 1.2837 - val_acc: 0.5604
Epoch 10/30
44/44 [==============================] - 2s 39ms/step - loss: 1.3464 - acc:
0.5193 - val_loss: 1.2246 - val_acc: 0.6285
Epoch 11/30
44/44 [==============================] - 2s 38ms/step - loss: 1.2726 - acc:
0.5847 - val_loss: 1.2043 - val_acc: 0.5913
Epoch 12/30
44/44 [==============================] - 2s 39ms/step - loss: 1.2027 - acc:
0.6015 - val_loss: 1.1915 - val_acc: 0.6192
Epoch 13/30
44/44 [==============================] - 2s 39ms/step - loss: 1.1785 - acc:
0.6150 - val_loss: 1.0630 - val_acc: 0.6811
Epoch 14/30
44/44 [==============================] - 2s 40ms/step - loss: 1.1238 - acc:
0.6113 - val_loss: 1.1539 - val_acc: 0.6254
Epoch 15/30
44/44 [==============================] - 2s 40ms/step - loss: 1.2021 - acc:
0.6292 - val_loss: 1.0540 - val_acc: 0.6502
Epoch 16/30
44/44 [==============================] - 2s 40ms/step - loss: 1.0318 - acc:
0.6582 - val_loss: 1.1621 - val_acc: 0.6285
Epoch 17/30
44/44 [==============================] - 2s 39ms/step - loss: 1.0688 - acc:
0.6574 - val_loss: 1.1716 - val_acc: 0.6192
Epoch 18/30
44/44 [==============================] - 2s 39ms/step - loss: 1.0489 - acc:
0.6525 - val_loss: 1.0344 - val_acc: 0.6656
Epoch 19/30
44/44 [==============================] - 2s 38ms/step - loss: 1.0067 - acc:
0.6690 - val_loss: 1.0394 - val_acc: 0.6811
```

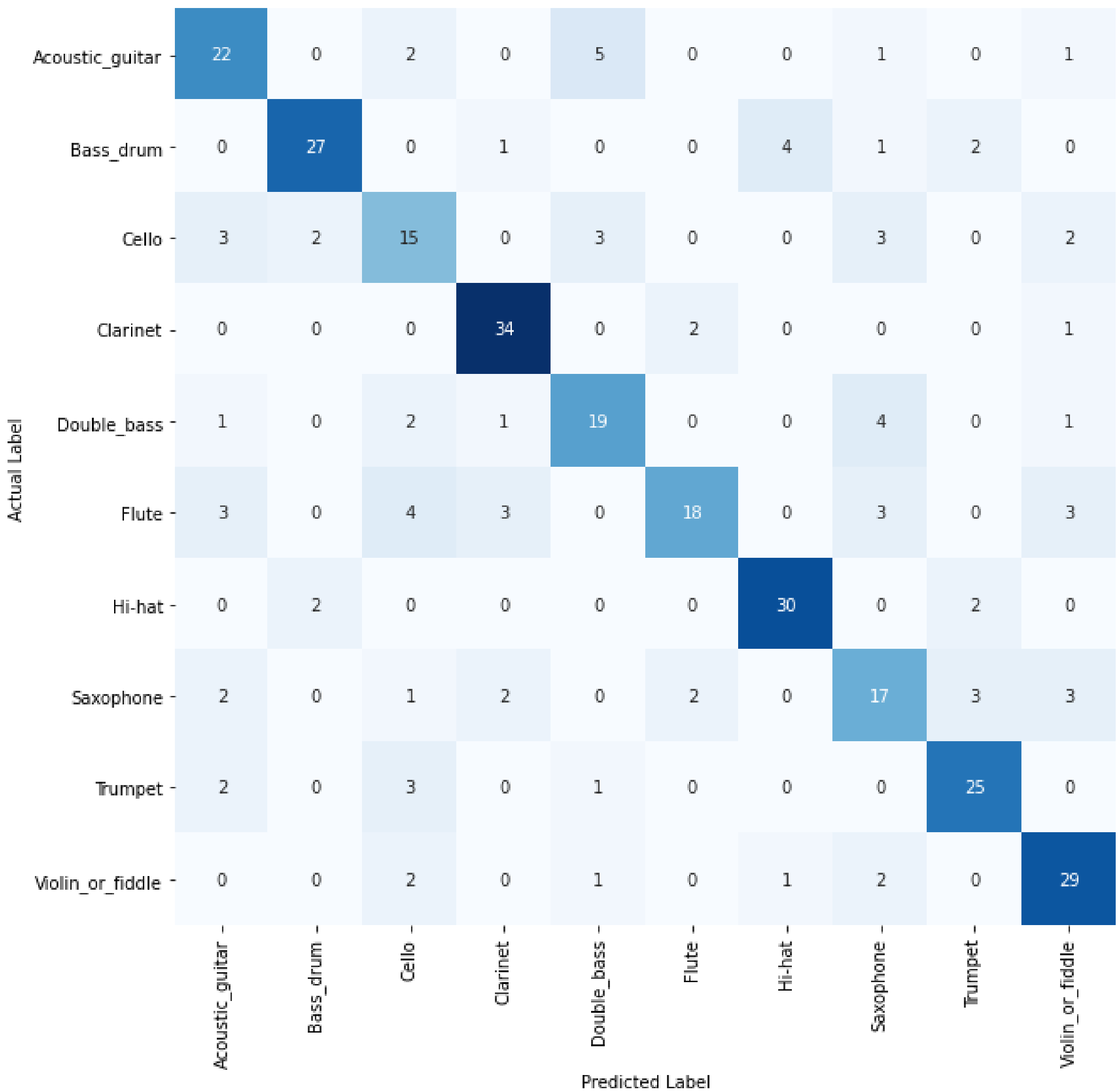
```
Epoch 20/30
44/44 [==============================] - 2s 39ms/step - loss: 0.9415 - acc:
0.7118 - val_loss: 0.9978 - val_acc: 0.7028
Epoch 21/30
44/44 [==============================] - 2s 39ms/step - loss: 0.9149 - acc:
0.7043 - val_loss: 1.0313 - val_acc: 0.6718
Epoch 22/30
44/44 [==============================] - 2s 39ms/step - loss: 0.9313 - acc:
0.6995 - val_loss: 1.0855 - val_acc: 0.6563
Epoch 23/30
44/44 [==============================] - 2s 40ms/step - loss: 0.9921 - acc:
0.6794 - val_loss: 1.0732 - val_acc: 0.6656
Epoch 24/30
44/44 [==============================] - 2s 39ms/step - loss: 0.9009 - acc:
0.6860 - val_loss: 1.0135 - val_acc: 0.6873
Epoch 25/30
44/44 [==============================] - 2s 39ms/step - loss: 0.8939 - acc:
0.7124 - val_loss: 1.0509 - val_acc: 0.6687
Epoch 26/30
44/44 [==============================] - 2s 40ms/step - loss: 0.8153 - acc:
0.7239 - val_loss: 0.9952 - val_acc: 0.6904
Epoch 27/30
44/44 [==============================] - 2s 39ms/step - loss: 0.8079 - acc:
0.7468 - val_loss: 0.9600 - val_acc: 0.6997
Epoch 28/30
44/44 [==============================] - 2s 39ms/step - loss: 0.8388 - acc:
0.7371 - val_loss: 0.9627 - val_acc: 0.6873
Epoch 29/30
44/44 [==============================] - 2s 39ms/step - loss: 0.7814 - acc:
0.7460 - val_loss: 0.9687 - val_acc: 0.6966
Epoch 30/30
44/44 [==============================] - 2s 40ms/step - loss: 0.8287 - acc:
0.7444 - val_loss: 0.9155 - val_acc: 0.7307
Loss: 0.8077050447463989
Validation Loss: 0.9155160784721375
```




```
Accuracy: 0.7422382831573486
Validation Accuracy: 0.7306501269340515
```



Accuracy
Accuracy
Validation Accuracy
0.7
0.6
0.5
0.4
0.3
0.2
0
5
10
15
20
25
30



Actual Label
Predicted Label
Acoustic_guitar
Bass_drum
Cello
Clarinet
Double_bass
Flute
Hi-hat
Saxophone
Trumpet
Violin_or_fiddle
22 0 2 0 5 0 0 1 0 1
0 27 0 1 0 0 4 1 2 0
3 2 15 0 3 0 0 3 0 2
0 0 0 34 0 2 0 0 0 1
1 0 2 1 19 0 0 4 0 1
3 0 4 3 0 18 0 3 0 3
0 2 0 0 0 0 30 0 2 0
2 0 1 2 0 2 0 17 3 3
2 0 3 0 1 0 0 0 25 0
0 0 2 0 1 0 1 2 0 29