Tim Northrop

October 7th, 2024

CS 4100 – Venkatesaramani

Final Project Report

Predicting Music Genre with the GTZAN Dataset

GitHub Repository: https://github.com/timnorthrop/cs4100_final_project

Abstract

The purpose of this project is to determine the most effective traits of audio input to use when identifying genre of music using neural networks. Specifically, this project investigates the performance of using mel spectrograms as input to a convolutional neural network (CNN) vs. using audio features in the form of raw data as input to a fully connected neural network (FCNN). Mel spectrograms are, per author Leland Roberts, “a way to visually represent a signal’s loudness, or amplitude, as it varies over time at different frequencies” [5]. I’ll be using a CNN for the spectrograms because of their aptitude at learning features at all scales with visual input, and an FCNN for the raw data input as the features extracted by the librosa package, included in the dataset, are already quite complex.

Introduction

I used to digitally produce music, experimenting with a broad range of genres and styles of music, and it’s always been fascinating to me how two songs that have great similarities can be classified as being in completely different genres, and similarly, how two songs that sound wildly different can be placed in the same genre. Because classification of a song’s genre is such a complex and subjective endeavor, I thought it would be interesting to investigate the effectiveness of neural networks in attempting to do so. I've investigated the performance of analyzing a visual representation of the audio vs. raw data extracted from the audio denoting data points like key, average pitch, BPM, etc. to get a better understanding of what it is that compels the general population to place songs in a certain genre.

Related Work

Many others have trained neural networks using the GTZAN dataset to accurately predict the genre of musical input, generate music, and more. Some interesting projects can be found here:

www.kaggle.com/code/andradaolteanu/work-w-audio-data-visualise-classify-recommend [2]

www.kaggle.com/code/dapy15/music-genre-classification [6]

www.kaggle.com/code/basu369victor/generate-music-with-variational-autoencoder [7]

Problem Statement and Methods

The goal here, once again, is not to predict genre as accurately as possible. It is to investigate which traits of a song determine its genre as we've decided it, and to do so without personal bias or opinion.

To accomplish this, I'll be using the GTZAN music genre classification dataset (also affectionately known as the MNIST of sounds [1]). The set consists of 1,000 .wav files (audio files) categorized and divided equally into 10 different genres. These genres are blues, classical, country, disco, hip-hop, jazz, metal, pop, reggae, and rock.

I’ve trained two models, GenreClassifierCNN and GenreClassifierFCNN, on inputs of mel spectrograms and audio features from CSV, respectively, and will go into more detail in the next section.

NOTE: The GTZAN dataset is deprecated, and my repository does not include code to retrieve the data. It can be downloaded from Kaggle here [1] and the contents of the “Data” folder can be placed locally in the “input” directory.

Experiments and Results

Firstly, to organize the data in a way that makes sense to our trainloaders and testloaders, I moved all mel spectrogram image examples from their respective genre folders up into the parent “images_original” folder to have a flattened list of all input.

I then defined two classes that extend Dataset: SpectrogramDataset and AudioFeatureDataset, each pointing to their respective data sources. SpectrogramDataset gets info from the “input/images_original” folder while AudioFeatureDataset gets its 58 normalized (with a StandardScaler from the sklearn package) features from the “input/features_30_sec.csv” file.

To use and test these two datasets, I defined two neural networks: GenreClassifierCNN and GenreClassifierFCNN. The CNN model has a very similar architecture to that of my submission for assignment 3, the fashion MNIST classifier, with 2 convolutional layers and two fully connected layers. The FCNN, because it would have significantly less tunable parameters with just 4 fully connected layers and to give it a fighting chance, uses the batch normalization function on the first two layers along with ReLU activation on the first three. As given by PyTorch documentation, the batch normalization algorithm is equivalent to the following [3]:

$y = \frac{x - E[x]}{\sqrt{Var[x] + e}} * \gamma + \beta$, where gamma and beta are "learnable parameter vectors whose size C is the number of features of the input" [3]. This algorithm is based on a study by Sergey Ioffe and Christian Szegedy [4].

```
class GenreClassifierCNN(nn.Module):
    def __init__(self):
        super(GenreClassifierCNN, self).__init__()
        self.conv1 = nn.Conv2d(3, 16, kernel_size=3, stride=1, padding=1)
        self.conv2 = nn.Conv2d(16, 32, kernel_size=3, stride=1, padding=1)
        self.pool = nn.MaxPool2d(2, 2)
        self.fc1 = nn.Linear(216 * 144 * 2, 128)
        self.fc2 = nn.Linear(128, 10)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = torch.flatten(x, 1)
        x = torch.relu(self.fc1(x))
        x = self.fc2(x)
        return x

class GenreClassifierFCNN(nn.Module):
    def __init__(self):
        super(GenreClassifierFCNN, self).__init__()
        self.fc1 = nn.Linear(58, 256)
        self.bn1 = nn.BatchNorm1d(256)
        self.fc2 = nn.Linear(256, 128)
        self.bn2 = nn.BatchNorm1d(128)
        self.fc3 = nn.Linear(128, 64)
        self.fc4 = nn.Linear(64, 10)

    def forward(self, x):
        x = F.relu(self.bn1(self.fc1(x)))
        x = F.relu(self.bn2(self.fc2(x)))
        x = F.relu(self.fc3(x))
        x = self.fc4(x)
        return x
```

I trained both networks over 30 epochs, using equally sized, randomized training sets. I then evaluated both on accuracy, the output of which can be seen here (ran twice):

```
CNN epoch 1 loss: 2.298168897628784
FCNN epoch 1 loss: 2.188281536102295
CNN epoch 2 loss: 2.176129102706909
FCNN epoch 2 loss: 1.7464852333068848
CNN epoch 3 loss: 2.0877842903137207
FCNN epoch 3 loss: 2.0708084106445312
CNN epoch 4 loss: 1.8830090761184692
FCNN epoch 4 loss: 1.7414318323135376
CNN epoch 5 loss: 1.8679137229919434
FCNN epoch 5 loss: 1.8543329238891602
CNN epoch 6 loss: 1.7399605512619019
FCNN epoch 6 loss: 1.8672288656234741
CNN epoch 7 loss: 1.46989905834198
FCNN epoch 7 loss: 2.0671730041503906
CNN epoch 8 loss: 1.5856901407241821
FCNN epoch 8 loss: 1.7574578523635864
CNN epoch 9 loss: 0.8798177242279053
FCNN epoch 9 loss: 1.6175341606140137
CNN epoch 10 loss: 1.263737440109253
FCNN epoch 10 loss: 1.692115068435669
CNN epoch 11 loss: 0.6865414977073669
FCNN epoch 11 loss: 1.5834214687347412
CNN epoch 12 loss: 0.6576858162879944
FCNN epoch 12 loss: 1.6347556114196777
CNN epoch 13 loss: 0.5741276741027832
FCNN epoch 13 loss: 1.6954466104507446
CNN epoch 14 loss: 0.312561571598053
FCNN epoch 14 loss: 1.694813847541809
CNN epoch 15 loss: 0.32594767212867737
FCNN epoch 15 loss: 1.5339076519012451
CNN epoch 16 loss: 0.1837979108095169
FCNN epoch 16 loss: 1.6154967546463013
CNN epoch 17 loss: 0.15169960260391235
FCNN epoch 17 loss: 1.597360372543335
CNN epoch 18 loss: 0.03742045536637306
FCNN epoch 18 loss: 1.6657061576843262
CNN epoch 19 loss: 0.06489866971969604
FCNN epoch 19 loss: 1.6273175477981567
CNN epoch 20 loss: 0.03228379786014557
FCNN epoch 20 loss: 1.574169635772705
CNN epoch 21 loss: 0.03818748891353607
FCNN epoch 21 loss: 1.7306296825408936
CNN epoch 22 loss: 0.02165772207081318
FCNN epoch 22 loss: 1.6221634149551392
CNN epoch 23 loss: 0.012716975063085556
FCNN epoch 23 loss: 1.5153098106384277
CNN epoch 24 loss: 0.03915422037243843
FCNN epoch 24 loss: 1.6360093355178833
CNN epoch 25 loss: 0.013647223822772503
FCNN epoch 25 loss: 1.9758087396621704
CNN epoch 26 loss: 0.010023129172623158
FCNN epoch 26 loss: 1.604527473449707
CNN epoch 27 loss: 0.007695438805967569
FCNN epoch 27 loss: 1.6359448432922363
CNN epoch 28 loss: 0.0036599377635866404
FCNN epoch 28 loss: 1.8510034084320068
CNN epoch 29 loss: 0.005172540433704853
FCNN epoch 29 loss: 2.0162811279296875
CNN epoch 30 loss: 0.002847518539056182
FCNN epoch 30 loss: 1.5705630779266357
CNN accuracy:  0.605
FCNN accuracy:  0.325
```

```
CNN epoch 1 loss: 2.299661874771118
FCNN epoch 1 loss: 1.9925639629364014
CNN epoch 2 loss: 2.1644837856292725
FCNN epoch 2 loss: 1.8152986764907837
CNN epoch 3 loss: 1.8565253019332886
FCNN epoch 3 loss: 2.0161237716674805
CNN epoch 4 loss: 1.8995842933654785
FCNN epoch 4 loss: 1.8232805728912354
CNN epoch 5 loss: 1.499338984489441
FCNN epoch 5 loss: 1.86757230758667
CNN epoch 6 loss: 1.2156492471694946
FCNN epoch 6 loss: 1.893725037574768
CNN epoch 7 loss: 1.2847670316696167
FCNN epoch 7 loss: 1.7332819700241089
CNN epoch 8 loss: 0.9406175017356873
FCNN epoch 8 loss: 1.7663719654083252
CNN epoch 9 loss: 1.0177456140518188
FCNN epoch 9 loss: 1.6615405082702637
CNN epoch 10 loss: 0.7582963705062866
FCNN epoch 10 loss: 1.8188395500183105
CNN epoch 11 loss: 0.5503278374671936
FCNN epoch 11 loss: 1.7202140092849731
CNN epoch 12 loss: 0.5144199728965759
FCNN epoch 12 loss: 1.7810670137405396
CNN epoch 13 loss: 0.4677411615848541
FCNN epoch 13 loss: 1.7052433490753174
CNN epoch 14 loss: 0.40954679250717163
FCNN epoch 14 loss: 1.8421058654785156
CNN epoch 15 loss: 0.237308531999588
FCNN epoch 15 loss: 1.8246264457702637
CNN epoch 16 loss: 0.18385805189609528
FCNN epoch 16 loss: 1.4365577697753906
CNN epoch 17 loss: 0.07526040077209473
FCNN epoch 17 loss: 1.7970173358917236
CNN epoch 18 loss: 0.08347124606370926
FCNN epoch 18 loss: 1.827272891998291
CNN epoch 19 loss: 0.053481973707675934
FCNN epoch 19 loss: 1.5776816606521606
CNN epoch 20 loss: 0.04919497296214104
FCNN epoch 20 loss: 1.4136494398117065
CNN epoch 21 loss: 0.03806464374065399
FCNN epoch 21 loss: 1.5358370542526245
CNN epoch 22 loss: 0.033972062170505524
FCNN epoch 22 loss: 1.8341134786605835
CNN epoch 23 loss: 0.018206385895609856
FCNN epoch 23 loss: 1.7372716665267944
CNN epoch 24 loss: 0.00854544062167406
FCNN epoch 24 loss: 1.5360839366912842
CNN epoch 25 loss: 0.007657856214791536
FCNN epoch 25 loss: 1.9470170736312866
CNN epoch 26 loss: 0.008032968267798424
FCNN epoch 26 loss: 1.6901800632476807
CNN epoch 27 loss: 0.008253299631178379
FCNN epoch 27 loss: 1.4021520614624023
CNN epoch 28 loss: 0.010722046718001366
FCNN epoch 28 loss: 1.9692251682281494
CNN epoch 29 loss: 0.01621604897081852
FCNN epoch 29 loss: 1.6605600118637085
CNN epoch 30 loss: 0.006980263162404299
FCNN epoch 30 loss: 1.4122439622879028
CNN accuracy:  0.575
FCNN accuracy:  0.28
```

Discussion and Conclusion

This experiment is not perfect. Both models can be more finely tuned to learn from their respective input types. However, it seems from the output of the program that the CNN using mel spectrograms as input is able to consistently and significantly outperform the FCNN using audio features in terms of accuracy. This speaks to both the complexity of the mel spectrogram as a tool and the advantages of using convolutional layers while working with visual input in neural networks. I’d be interested to continue this investigation to find out if a more complex neural network can be used to work with the audio features to approach the CNN’s level of accuracy with spectrograms. This might further support the inference that can be (tentatively) made from this experiment: that mel spectrograms are a better way to gauge a song’s genre than raw data containing audio features.

Resources

1) Andrada. “GTZAN Dataset - Music Genre Classification.” *Kaggle*, 24 Mar. 2020, www.kaggle.com/datasets/andradaolteanu/gtzan-dataset-music-genre-classification/data.
2) Andradaolteanu. “Work W/ Audio Data: Visualise, Classify, Recommend.” *Kaggle*, Kaggle, 25 Mar. 2020, www.kaggle.com/code/andradaolteanu/work-w-audio-data-visualise-classify-recommend.
3) PyTorch Documentation. 2 Mar. 2015 https://pytorch.org/docs/stable/generated/torch.nn.BatchNorm1d.html.
4) Sergey Ioffe, Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift." (2015)
5) Roberts, Leland. “Understanding the Mel Spectrogram.” Medium, Analytics Vidhya, 17 Jan. 2024, www.medium.com/analytics-vidhya/understanding-the-mel-spectrogram-fca2afa2ce53.
6) dapy15. “Music Genre Classification.” Kaggle, Kaggle, 14 May 2021, www.kaggle.com/code/dapy15/music-genre-classification.
7) basu369victor. “Generate Music with Variational AutoEncoder.” Kaggle, Kaggle, 27 Dec. 2021, www.kaggle.com/code/basu369victor/generate-music-with-variational-autoencoder.