**Music Models classification: Contigo & BetterBots**

Ambatipudi Abhiram (11840170) Ayush Gupta (11840290)

Asad Abidi (11840220) Shalini Kumari (11841030)

**Abstract :**

Due to the advent of machine learning, many problems were solved, automated, and made easier but a few problems like the problem of deepfakes were created. We are making a detector system to classify among different music generating machine learning models. The aforementioned is a state-of-the-art problem and there hasn’t been much research done in this area.

We created our own dataset having music generated by popular and available Machine Learning models. Converted the music files into spectrogram format (which is the visual representation of sound). Then our problem was reduced into a multi-class classification problem. We analyzed different available models for the same. While there were different models that we might have used, we used the ResNet50 model since it had fairly high accuracy.

With our naive model, we got the highest accuracy of 57% on training data and 39% on validation data. On the other hand, after doing transfer learning with ResNet50 our accuracy improved significantly, with training accuracy of 66% and validation accuracy of 52%. (after running for 200 epoch due to GPU constraints)

We have achieved adequate results considering our limited time and resources. Moving in the same direction and using even newer and complicated models we may presumably achieve higher accuracy.

**Introduction and Problem Definition :**

Whether one wants to create a fake video of something or someone or clone someone's voice, machine learning models are very efficient in doing all these. Nowadays Machine Learning can generate faces of humans that never existed. So it’s quite difficult for a non-technical person to distinguish between what’s real or what’s fake. Fraudulent people may try to scam a person by cloning the voice of their relatives or friends. But there can only be a limited number of popular ML models for the scammers to use (since it is of very low possibility that they have in-depth knowledge of machine learning to create new models).

Our goal is to train a model to find out whether the music sound taken as input by our model is created by any of the known and popular machine learning models out there in the market.

As there are moral and ethical dilemmas associated with the use of many AI systems. We hereby represent a system that a community can use to differentiate between what’s real music and what’s fake.

**Objective :**

Our goal is to develop a system that, upon taking input as a music file, predicts from which machine learning model this music has been generated. Currently, our system is limited to three models only namely basic RNN, Loopback RNN, and attention RNN.

**Technology Used :**

Libraries used

1. Python
2. numpy
3. matplotlib
4. pandas
5. keras
6. Magenta
7. Tensorflow
8. RoX
9. Random

**Problems Faced :**

1. This problem was scantily explored before in research labs of Machine learning and artificial intelligence. And it was difficult to look into various aspects of audio models which spanned from basic RNN to attention RNN.
2. We didn't previously have domain knowledge about our problem statement. We had to first acquire all the information about the three models used. Then we had to search for the best model suited for our problem statement.
3. Keeping the track of global features and local features at the same time was a difficult task as some models were based upon global structure throughout their melodies while others were based upon local structure.
4. Keeping in mind all this we started to look into the problem from scratch and observed a scarcity of resources online. Given more time and resources, we could have made a better model.
5. Also, this is not the kind of problem in which we can improve our accuracy with human introspection and data augmentation.
6. There can be no concept called human-level performance in this problem. Because if we were to give someone three music audios generated by three different models to listen to and identify which model was used to generate which audio, they probably wouldn't be able to do it.

**Datasets Used :**

There were no open-source datasets available for this project. So we decided to create our own dataset and contribute to the open-source community. To start with we looked into the magenta library supported by TensorFlow. We found various models for the music generation present there. As our team previously worked upon some of them, we chose the following models:

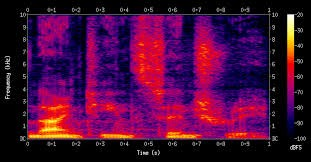
* Attention RNN
* Lookback RNN
* Basic RNN

We used a random library of python to create different primer sequences for our models. By feeding these primer sequences into the model we created 600 music files from each model which were in MID format. But we needed the data in WAV format, so we used a Linux file converter **TiMidity** to convert it into the WAV format which turned out to be around 4.5 GB.

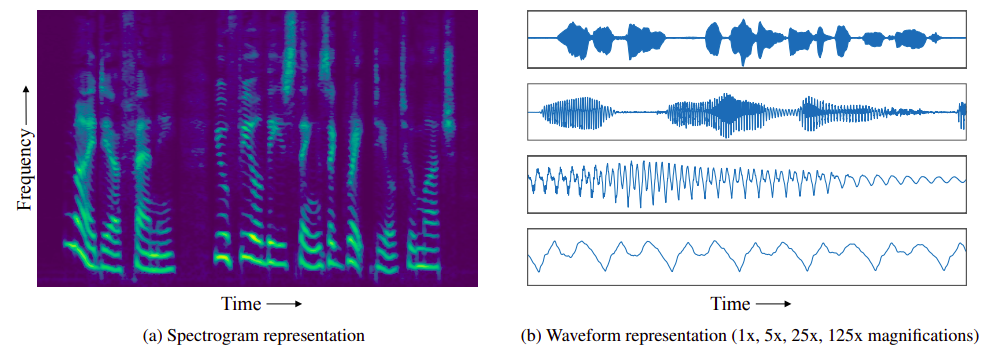
We used another Linux command-line utility **SoX**,whichis a cross-platform command-line utility that can convert various formats of computer audio files into other formats to convert the WAV files into corresponding **spectrograms** (which are a visual way of representing the signal strength, or loudness, of a signal over time at various frequencies present in a particular waveform.)

**Preprocessing :**

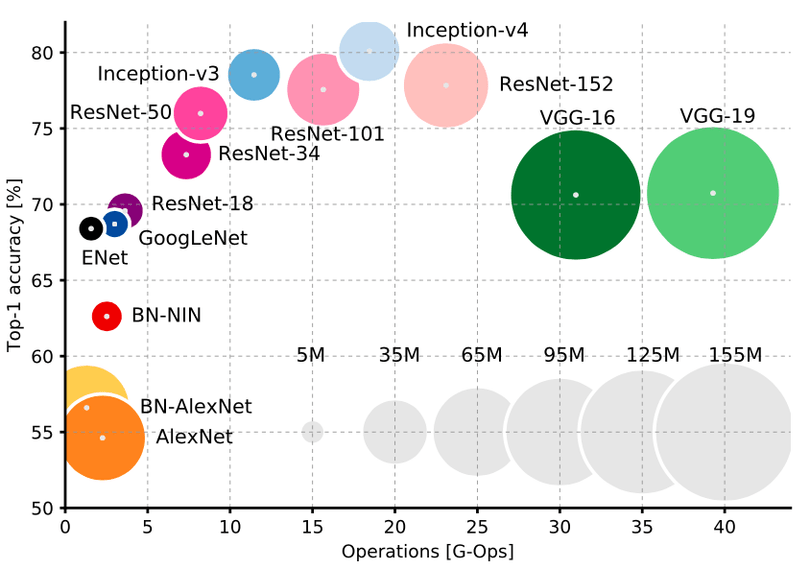
Capturing high-level structure in audio waveforms is challenging because a single second of audio spans tens of thousands of timesteps. While long-range dependencies are difficult to model directly in the time domain, we choose two-dimensional me-frequency representations of the audio file namely **spectrograms**. Modeling spectrograms can simplify the task of capturing global structure but can weaken a model’s ability to capture local characteristics.

****

A **spectrogram** is a visual way of representing the signal strength, or “loudness”, of a signal over time at various frequencies present in a particular waveform. Not only can one see whether there is more or less energy at, for example, 2 Hz vs 10 Hz, but one can also see how energy levels vary over time. [Spectrograms](https://en.wikipedia.org/wiki/Spectrogram) function more or less like heatmaps for sound and are closely related to a more common visual representation of sound called waveforms.



**Methodology:**

After doing all preprocessing steps we reached a multi-class classification problem. So we analyzed different available models for the same. While there are different models that we might use, the **ResNet50** model has fairly high accuracy, while not being too big in comparison to the computationally expensive VGG and other network architecture. So we opted for ResNet50.****

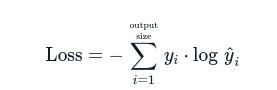
We used Adam optimizer for optimization of our model and categorical cross-entropy loss as loss function. The brief info about both of them is given in the following section. Although we have tried RMSProp it didn’t work well with our problem.

**Adam optimizer:**

Adam is an optimization algorithm that can be used to update network weights iteratively based on training data. We describe Adam as combining the advantages of two other extensions of stochastic gradient descent. Specifically:

* Adaptive Gradient Algorithm (AdaGrad) that maintains a per-parameter learning rate that improves performance on problems with sparse gradients.
* Root Mean Square Propagation (RMSProp) also maintains per-parameter learning rates that are adapted based on the average of recent magnitudes of the gradients for the weight (e.g. how quickly it is changing). This means the algorithm does well on online and non-stationary problems (e.g. noisy).

**Categorical Cross-Entropy loss**: Categorical cross-entropy is a loss function that is used in multi-class classification tasks. These are tasks where an example can only belong to one out of many possible categories, and the model must decide which one. The categorical cross-entropy loss function calculates the loss of an example by computing the following sum:



Where yi cap  is the *i*-th scalar value in the model output, yi is the corresponding target value, and output size is the number of scalar values in the model output.

**Model Used :**

After doing all preprocessing with the dataset, the only part remaining was to use a suitable classification model according to our dataset. For that, we used two models. First was a simple model created using sequential model layers from Keras.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

conv2d (Conv2D) (None, 298, 298, 32) 896

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

batch\_normalization (BatchNo (None, 298, 298, 32) 128

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d (MaxPooling2D) (None, 149, 149, 32) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout (Dropout) (None, 149, 149, 32) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_1 (Conv2D) (None, 147, 147, 64) 18496

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

batch\_normalization\_1 (Batch (None, 147, 147, 64) 256

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_1 (MaxPooling2 (None, 73, 73, 64) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_1 (Dropout) (None, 73, 73, 64) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_2 (Conv2D) (None, 71, 71, 128) 73856

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

batch\_normalization\_2 (Batch (None, 71, 71, 128) 512

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_2 (MaxPooling2 (None, 35, 35, 128) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_2 (Dropout) (None, 35, 35, 128) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

flatten (Flatten) (None, 156800) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense (Dense) (None, 512) 80282112

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

batch\_normalization\_3 (Batch (None, 512) 2048

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_3 (Dropout) (None, 512) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_1 (Dense) (None, 3) 1539

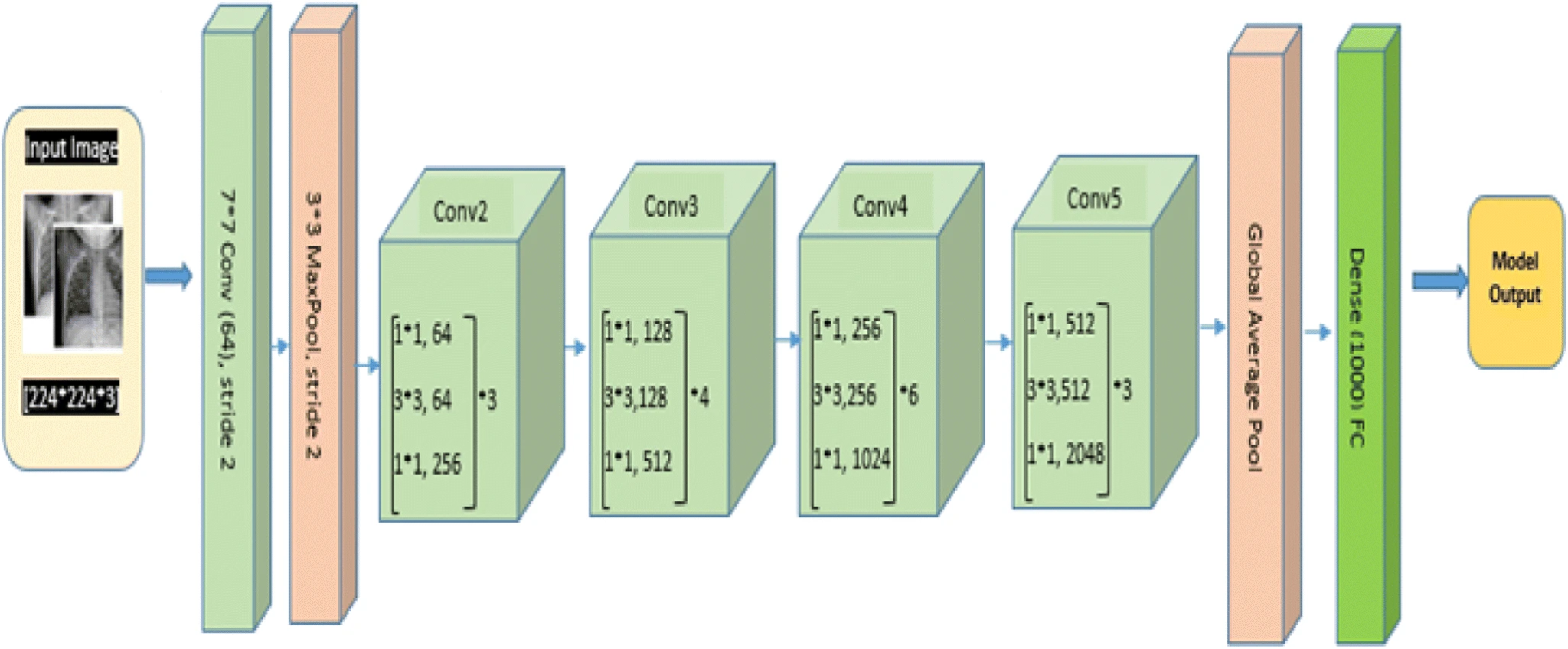
=================================================================

Total params: 80,379,843

Trainable params: 80,378,371

Non-trainable params: 1,472

Another one was ResNet50 which we didn’t train from scratch but did transfer learning. **ResNet50** is a convolutional neural network that is 50 layers deep. Which was first introduced by Microsoft for the Imagenet challenge for multi-class classification problems which includes 1000 categories of a wide range. There were other models available too but we chose ResNet50 as it solves the vanishing gradient problem which generally arises with very deep neural networks, with the help of skip connections. You can see the architecture of ResNet in the following figure. For more detailed information see [here](https://cdn-5f733ed3c1ac190fbc56ef88.closte.com/wp-content/uploads/2019/07/ResNet50_architecture-1.png).



We simply removed the last fully connected layer and added another fully connected layer with 3 sigmoid outputs for our classification purpose.

**Results and Performance :**

With our naive model, we got the highest accuracy of 57% on training data and 39% on validation data, after running for 200 epochs. On the other hand, after doing transfer learning with ReSNet50 our accuracy improved significantly, with training accuracy of 66% and validation accuracy of 52%.

**Conclusion :**

We have generated a classification model which can reasonably identify which model was used while generating a particular music file in the given time constraints.

**Future Work :**

1. We will try to look into **R-CNN** based models because most of the models used for generating data were based upon RNN. We think that R-CNN based models would give a better state-of-the-art performance.
2. We will try to include all the open-source models available for our classification task.

**References:**

1. [**https://arxiv.org/pdf/1906.01083.pdf**](https://arxiv.org/pdf/1906.01083.pdf)
2. [**https://www.ijstr.org/final-print/oct2016/Detection-Of-Alterations-In-Audio-Files-Using-Spectrograph-Analysis.pdf**](https://www.ijstr.org/final-print/oct2016/Detection-Of-Alterations-In-Audio-Files-Using-Spectrograph-Analysis.pdf)
3. [**https://towardsdatascience.com/understanding-and-coding-a-resnet-in-keras-446d7ff84d33**](https://towardsdatascience.com/understanding-and-coding-a-resnet-in-keras-446d7ff84d33)
4. [**https://medium.com/dessa-news/detecting-audio-deepfakes-f2edfd8e2b35**](https://medium.com/dessa-news/detecting-audio-deepfakes-f2edfd8e2b35)