January 12th, 2022

The Final Paper: Effect of Different Data Types on the Accuracy of Machine Learning Algorithms

S. Ojha, N. Nakovick, A. Babbin

High Technology High School, 765 Newman Springs Road, Lincroft, NJ 07738

Abstract

Classification algorithms are a subset of machine learning (ML) programs that are designed to group pieces of data together based on characteristics that they identify in the data over time. The subject of this paper involves the testing of a classification algorithm with either image data or audio data, the independent variable of the experiment, to see if there is a significant difference between how well either of the data types train the algorithm and how accurate the algorithms end up being with regards to how many pieces of data it can classify correctly, with this being the dependent variable. It has been hypothesized that there would be a significant difference between the image and audio data's performance. To do this, two datasets are curated – one containing only images of cats and dogs, and the other containing spectrograms of meowing/barking sounds made by cats and dogs. These datasets are then individually uploaded to a classification algorithm on Kaggle, through which the algorithms can train and test themselves. The final testing accuracy is then recorded over the course of 24 total trials (12 for both types of data), and analyzed to see whether there is a significant difference between how well either performed. The results have shown that there is a significant difference between the audio data and the image data's performance. In the future, studies researching this topic will use larger datasets and novel scraping methods to acquire data to put in the datasets, while also addressing the problem of how repeated occurrences of the same data in training over other pieces of data may skew the accuracy of the final model.

Introduction

Data scientists use a variety of tools and techniques to help them analyze social, financial, or scientific trends. Machine learning (ML) algorithms, which use large amounts of data to make predictions, are one such tool. In ML, a classification algorithm's purpose is to sort data into different groups. Studies have been conducted on these algorithms since their inception, allowing artificial intelligence researchers to make generalizations about the way they work. In general, they have found that the results of these algorithms depend on whether the data is labeled or unlabeled, since labeled data, which often has to be prepared by scientists and can even cost money, can be used in ML algorithms to sort data into predetermined categories based on the labels, while unlabeled data cannot, but is easier to come by because of its ability to be scraped, or taken in large quantities from the internet (Blum and Mitchell). In addition, structured data, such as data found in tables and organized sheets, are analyzed faster and more easily than unstructured data, such as the unorganized data in images or chunks of text (Sarker). This study looked at how the use of different types of unstructured data – audio and image data – affected the accuracy of a classification algorithm. With both audio and images, the algorithm would be tasked with classifying the data into either a cat group or a dog group.

In the past, to write ML classification algorithms for unstructured data, some tools were necessary. Many data analysts have written their algorithms in the programming language Python. This language has been used in ML due to its easy-to-read structure and access to libraries, or sets of pre-written code for common tasks. Specifically, to work with images, data analysts have needed to familiarize themselves with convolutional neural networks (CNNs), which were designed to learn like humans by being fed chunks of image data. As more data would be fed, the CNN would create feature maps, which allowed an algorithm to turn each pixel into a number, quantifying its appearance, and compare these numbers between images to generate trends over time. These feature maps have gotten better as the use of CNNs has increased (Chakraborty et al.).

CNNs have been used in everything from surveillance to acoustic system analysis, and have also been extremely commonly used in image and audio processing/classification (Ajmera and Sinha). Recent work has also brought up the use of spectrograms, or images displaying a spectrum of frequencies, made from audio files to classify them through CNNs (Fu et al.). In particular, many modern CNNs have been geared towards the recognition and analysis of living organisms. One study used CNNs to classify plants based on whether they appeared to be diseased or not (Mohanty et al.). Another similar study used CNNs to identify insects the moment they come into the field of view of a laser device that attacks them, serving as pest control (Silva et.al.). Since more information and data exists online for images and audio of cats and dogs, these will be used in the study.

Within the scope of ML research, this study's results could help data scientists make more distinctions regarding the different types of data that can be used in a classification algorithm. While the differences in structured and unstructured data are clearly outlined, it is not fully known how different types of unstructured data behave in the neural networks being used in this study. Many of the procedures involved in analyzing image and audio data are the same, in that an audio file is converted into an image of its spectrogram, which could then be analyzed through a CNN. Therefore, this study could shed light on how the two differ in terms of accuracy. Based on this information, choices about which data type will provide the most accurate results could be made easier.

Based on the conditions of this experiment, it was hypothesized that there was a significant difference between the accuracy of the image data and the audio data. The primary reason for this hypothesis was that the audio data found for cats and dogs was highly variable. This was generally evidenced by the continuously-changing length of the audio file and the pitch of the animal's sounds. Because of this, all of the spectrograms created for the audio files would look extremely different, making similarities much harder to compare, if not incomparable. On the other hand, since all of the pictures had to be the same size from the get-go, any variability between pictures was lessened. Due to these differences, it was predicted that image data and audio data would have large differences in their accuracy. As it stood, *the alternative*

hypothesis was that there is a significant difference between the accuracies of image data and audio data, and the null hypothesis was that there is NOT a significant difference between the accuracies of image data and audio data.

To collect data to prove or disprove this hypothesis, this study involved the execution of two algorithms trying to sort data split between cats and dogs. The independent variable in the study was the use of either 1000 files worth of image or audio data. Both image and audio data was preprocessed as images, with the audio being converted into spectrograms showing the frequency of the sound. As per the procedure of a CNN, this data was split up into groups in which the algorithm could first be trained and then tested to see whether it could differentiate between cats and dogs. The independent variable in this case was the end accuracy of each algorithm, which was recorded over the course of 12 trials.

Materials and Methods

Preparing Materials

There were not many physical materials needed to conduct this study. The only physical object needed to conduct this experiment was a computer, which ran the ML algorithms that are being tested in this study. Since these algorithms had to work through large amounts of data to work properly, this computer had to have been fairly powerful. The one used in this study was a Lenovo Yoga C940, running on an Intel® Core™ i7 10th Gen CPU and an Intel® Iris® Plus Graphics GPU. All other materials were found on the computer itself. This study necessitated the use of Python, a general-purpose programming language often used in data science and ML.

To run properly, ML algorithms work best in Ipython notebooks (denoted by the .ipynb file type), given large amounts of data. Kaggle was used to address both of these concerns. Kaggle provides free access to datasets, which will be used in the algorithms to train the models (Carpita et al.). Once a dataset is chosen through Kaggle, the program allows one to immediately open a Python notebook in which the algorithm can be written and tested. This made conducting the study more convenient.

The code being written in Python also means that the code had access to many different libraries. First, the os library will be needed to access the directory of the notebook, allowing for the manipulation of data loaded into the notebook. After this, the tensorflow library, which is a popular ML library made by Google, will do most of the ML work (Abadi et.al.). Finally, numpy will generate random numbers to unbatch and visualize the data and if the algorithm has learned how to classify it, while scipy will be used to quantify the attributes of audio files.

Before setting up and conducting the study, some experimental guidelines were needed to obtain untainted and unbiased results. The control group in this study was an algorithm that attempted classification with image data, while the experimental group was an algorithm that attempted classification with audio data. Constants also needed to be defined as well to ensure the best results. For one, both ML algorithms had to run at the same time and in the same place. Due to some of the WiFi problems

that the school has faced at times, along with the fact that a computer cannot be kept open in school at all times, all the trials had to be conducted at home. To ensure that the computer was running at its maximum capacity, all tabs except for those open on the Ipython notebook were closed, and the computer was plugged in at all times, keeping it 100% charged. Finally, both algorithms being tested used the same code, since having different code could have changed the testing and training data enough with every trial that the results may have been skewed. Therefore, the audio files were converted into spectrograms before being uploaded to the notebooks in Kaggle. These constants were essential in providing the best possible results towards the study. The ultimate goal was to see how audio classification differed in terms of accuracy from the image control.

Preparing the Datasets

In ML, a dataset is a large amount of data that can be analyzed by an algorithm. Two of these needed to be created to test the algorithm used in this experiment. The first dataset consisted of pictures of cats and dogs, making up the image control group of the experiment, while the second dataset started off consisting of recordings of cats and dogs making sounds, but was converted to spectrograms. Since it was considerably tougher to find reliable audio files of the sounds made by cats and dogs, the dataset was capped at 1000 audio files, and had to remain the same for the image files as well.

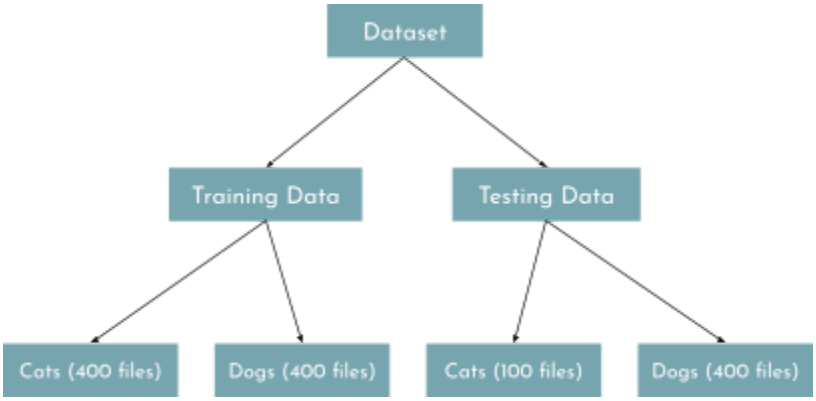

Figure 1: The training and testing folders correspond to what role the files served, as data in the training folder was used to teach the algorithm to recognize patterns, while data in the testing folder was thrown at the algorithm to see if it could consistently recognize the same patterns it was taught previously.

The image dataset did not require as much work to make, since large datasets with images of cats and dogs already existed. The one that was used in this project was derived from the *Dogs & Cats Images* dataset by chetanimravan on Kaggle. This dataset had 10000 images, with 4000 images each in the training folders and 1000 each in the testing folders. In each folder, 90% of the images were pruned from the dataset to make sure that the final dataset had a total of 1000 images, with 400 each in the training folders and 100 each in the testing folders. This could easily be done by downloading the dataset, deleting the images manually, and uploading the remaining images as a new dataset on Kaggle. The dataset I would end up making was called *Tiny Cats and Dogs Images Data*, and has been made public on Kaggle.

Making the audio dataset required more work, since numerous sources were needed to assemble a full dataset. The files had to be in .wav format, since these usually have higher quality audio than .mp3 files. To get 1000 audio files, sound was taken from CatMeows, containing 440 files of cats making sounds. This leaves 60 audio files needed for the cats. This was taken from the *Audio Cats and Dogs* dataset by Marc Moreaux, which has 162 cat audio files – any extra cat files could be discarded at this stage. To acquire dog data, Marc Moreaux's dataset had 117 audio files for dogs. BarkMeowDB, which can be found on Zenodo, had 46 recordings of barking dogs. Finally, Google's Audioset service provided 2,632 videos of dogs barking. Of these, all 120 evaluation and balanced train videos, as well as the first 217 unbalanced train videos, were converted into wav files through online services such as online-convert.com. This ensured that there are 500 audio files of dogs barking, which completed the dataset.

To ensure that the code used in both image and audio analysis remained consistent, and to analyze the audio correctly, all of the .wav files in the audio dataset will be converted into spectrograms. This can be done by using the scipy library to read the file and find attributes such as the amount of time the audio ran and the

frequencies of the audio. After this, matplotlib can be used to graph the final spectrogram, which can be used in the classification algorithm. The code, as well as an example of one spectrogram, can be seen below.

```
import matplotlib.pyplot as plt
from scipy import signal
from scipy.io import wavfile
import os
from pathlib import Path

catnum = 0

for cat in Path("../archive (4)/dataset/test_set/cats/").glob("*.wav"):
    catnum += 1

    sample_rate, samples = wavfile.read(cat)
    frequencies, times, spectrogram = signal.spectrogram(samples, sample_rate)

    plt.pcolormesh(times, frequencies, spectrogram)
    plt.imshow(spectrogram)
    plt.ylabel('Frequency [Hz]')
    plt.xlabel('Time [sec]')
    plt.show(block=False)
    plt.savefig(f"cat_testing_{catnum}")
    plt.pause(1)
    plt.close()
```

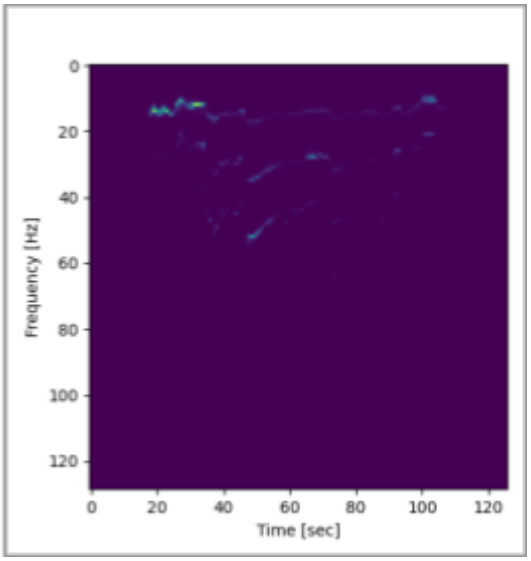

Figure 2: The code above saved all of the files in the cat_testing dataset as spectrograms instead of wav files. This is what one of them looked like.

Writing the Algorithms

To set up the study, two algorithms were written. One used a dataset of spectrograms of the sounds of barking dogs or meowing cats. The other used a dataset of images that showed either a cat or a dog. Both would be tasked with differentiating between the two. Both were supervised learning algorithms. These are essentially ML programs that are given a set of data to learn from, as well as the labels they are given. For example, an image in the dataset would either be labeled a cat or a dog. This acts as an answer key of sorts, forcing the algorithm to choose between the cat or the dog, and helping it identify the traits of either entity. With audio, the file would either be labeled cat or dog, and just like the image data, the algorithm would be forced to choose between either cat or dog, learning their unique traits in the process.

The algorithms in this study did not require any physical assembly. Instead, the algorithms were coded on the computer. To start, the datasets were imported into the Python notebook on Kaggle. This was done by opening the notebook and clicking the “Add Data” button in the top right corner. This gave the notebook access to all of the data that was to be used throughout the study. After this, the packages involved in this algorithm had to be imported into the notebook. The researcher could then access each of the files in the code by using the os library to determine the path, or the text indicating the location of each file in the notebook. This allowed the algorithm to split up and access the data later.

```
[ ]:
# loading the training data
cats_path = os.listdir("../input/catdog-audio-data/archive (4)/dataset/training_set/cats/")
dogs_path = os.listdir("../input/catdog-audio-data/archive (4)/dataset/training_set/dogs/")
cats_full_path = ["../input/catdog-audio-data/archive (4)/dataset/training_set/cats/" + fname for fname in cats_path]
dogs_full_path = ["../input/catdog-audio-data/archive (4)/dataset/training_set/dogs/" + fname for fname in dogs_path]
X = cats_full_path + dogs_full_path
cats_labels = [0]*len(cats_full_path)
dogs_labels = [1]*len(dogs_full_path)
y = cats_labels + dogs_labels
```

Figure 3: The os library takes the path of the training files and calculates how many labels there are.

Before training the algorithm, the images in each dataset first needed to be preprocessed. For this, tensorflow was imported, after which each image in the dataset was accessed through its path, read pixel-by-pixel, and had its label returned so that it could be attributed to a specific category when needed for testing.

When working in a supervised learning algorithm, data was split up unevenly into training and testing data. The bigger portion, training data, provided the algorithm with an idea of the traits of each unique image. In image data, a CNN, or convolutional neural network, is used to create a feature map and analyze an image pixel by pixel. A number is assigned to each pixel in the image, and with every image that is analyzed by the CNN, the numbers are assigned weights and compared to see whether the same numbers overlap over pixels. This allows the CNN to identify patterns in the images, and classify the images that way.

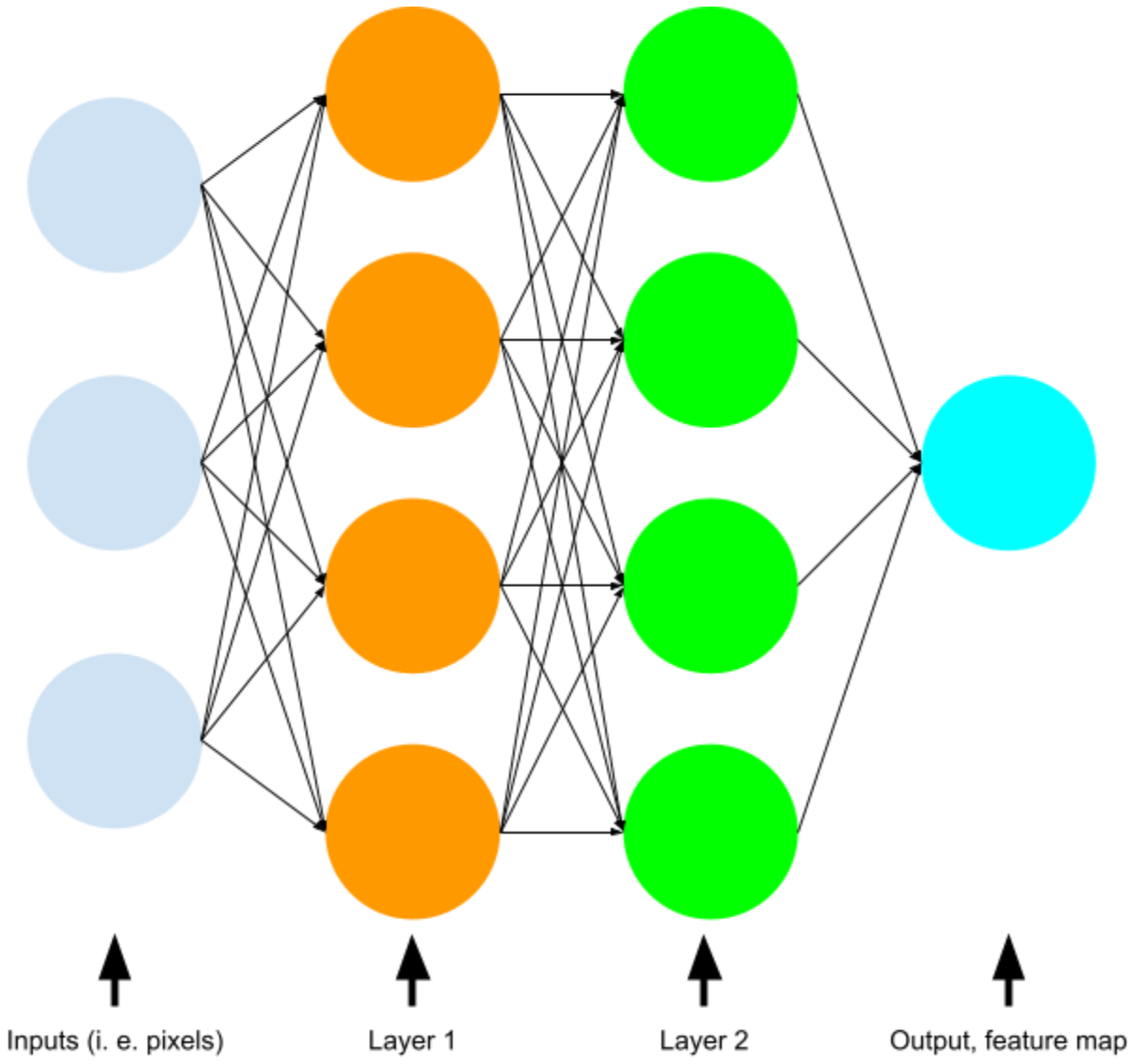

Figure 4: A simple convolutional neural network. Each pixel of an image was inputted into multiple layers that assign them different weights. As time goes on, these pixels will start to show patterns in their placement and weight values. This creates a feature map that can be used to classify the images.

The same holds true for the audio files, as they were converted into spectrograms and curated into a dataset prior to the execution of the algorithm. Once all the training data was exhausted, the algorithm switched to a smaller subset of data, or testing data. The algorithm then saw how many different labels it can identify, and quantified its accuracy as a percentage. Once testing data was run through completely, new training data was given, and the cycle repeated. This process was layered as

many times as necessary, and helped make the program more accurate over time. This process is known as training a model.

```
# function which builds our model
def create_model():
    # building the model
    model = tf.keras.models.Sequential([
    # building our CNN
    tf.keras.layers.Conv2D(16, (3,3), activation='relu', input_shape=(IMG_SIZE, IMG_SIZE, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),

    tf.keras.layers.Conv2D(32, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),

    tf.keras.layers.Conv2D(32, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),

    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),

    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),

    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),

    # Flatten the results to feed into a DNN
    tf.keras.layers.Flatten(),
    # 512 neuron hidden layer
    tf.keras.layers.Dense(512, activation='relu'),
    # Only 1 output neuron. It will contain a value from 0-1 where 0 for 'Cat' and 1 for 'Dog'
    tf.keras.layers.Dense(1, activation='sigmoid')
    ])

    model.compile(
        loss=tf.keras.losses.binary_crossentropy,
        optimizer=tf.keras.optimizers.Adam(),
        metrics='accuracy'
    )

    return model
```

Figure 5: The image data's feature map was multiplied and compared through Conv2D. This would be fed into a 512-neuron layer that would help assign weights to pixels and find commonalities in the images. This model was created through tensorflow's Keras library, which works as an interface between Python and tensorflow.

```python
# function to train and return a trained model
def train_model():
    """
    Trains a given model and returns the trained version.
    """
    # Create a model
    model = create_model()

    history = model.fit(
        train_data,
        epochs=NUM_EPOCHS,
        validation_data=val_data,
        callbacks= [early_stopping])

    return model, history
```

Figure 6: The model was forced to train. Epochs refer to the number of times it will cycle through testing data. This was set to 20.

```
Epoch 1/20
2022-01-11 17:18:17.512381: I tensorflow/stream_executor/cuda/cuda_dnn.cc:369] Loaded cuDNN version 8005
20/20 [==============================] - 12s 228ms/step - loss: 0.6984 - accuracy: 0.4750 - val_loss: 0.6927 - val_accuracy: 0.5000
Epoch 2/20
20/20 [==============================] - 1s 66ms/step - loss: 0.6936 - accuracy: 0.4750 - val_loss: 0.6928 - val_accuracy: 0.5000
Epoch 3/20
20/20 [==============================] - 1s 68ms/step - loss: 0.6934 - accuracy: 0.4750 - val_loss: 0.6925 - val_accuracy: 0.5000
Epoch 4/20
20/20 [==============================] - 1s 68ms/step - loss: 0.6933 - accuracy: 0.4922 - val_loss: 0.6914 - val_accuracy: 0.5000
Epoch 5/20
20/20 [==============================] - 1s 65ms/step - loss: 0.6931 - accuracy: 0.5125 - val_loss: 0.6897 - val_accuracy: 0.5000
Epoch 6/20
20/20 [==============================] - 2s 78ms/step - loss: 0.6910 - accuracy: 0.5156 - val_loss: 0.6870 - val_accuracy: 0.5875
Epoch 7/20
20/20 [==============================] - 1s 68ms/step - loss: 0.6873 - accuracy: 0.5625 - val_loss: 0.6897 - val_accuracy: 0.5063
Epoch 8/20
20/20 [==============================] - 1s 66ms/step - loss: 0.6682 - accuracy: 0.6156 - val_loss: 0.6867 - val_accuracy: 0.5437
Epoch 9/20
20/20 [==============================] - 1s 66ms/step - loss: 0.6643 - accuracy: 0.5734 - val_loss: 0.7416 - val_accuracy: 0.5625
Epoch 10/20
20/20 [==============================] - 1s 71ms/step - loss: 0.6485 - accuracy: 0.6031 - val_loss: 0.6863 - val_accuracy: 0.6500
Epoch 11/20
20/20 [==============================] - 2s 78ms/step - loss: 0.5921 - accuracy: 0.6906 - val_loss: 0.7395 - val_accuracy: 0.6125
Epoch 12/20
20/20 [==============================] - 1s 66ms/step - loss: 0.5497 - accuracy: 0.7141 - val_loss: 0.8048 - val_accuracy: 0.5938
Epoch 13/20
20/20 [==============================] - 1s 69ms/step - loss: 0.4844 - accuracy: 0.7672 - val_loss: 0.9352 - val_accuracy: 0.5625
Epoch 14/20
20/20 [==============================] - 1s 68ms/step - loss: 0.4147 - accuracy: 0.8078 - val_loss: 0.7675 - val_accuracy: 0.6438
Epoch 15/20
20/20 [==============================] - 1s 67ms/step - loss: 0.3689 - accuracy: 0.8406 - val_loss: 0.9084 - val_accuracy: 0.6062
Epoch 16/20
20/20 [==============================] - 1s 66ms/step - loss: 0.3391 - accuracy: 0.8438 - val_loss: 0.9148 - val_accuracy: 0.6062
Epoch 17/20
20/20 [==============================] - 1s 64ms/step - loss: 0.2168 - accuracy: 0.9094 - val_loss: 1.2074 - val_accuracy: 0.6187
Epoch 18/20
20/20 [==============================] - 1s 65ms/step - loss: 0.2051 - accuracy: 0.9062 - val_loss: 1.1473 - val_accuracy: 0.6375
Epoch 19/20
20/20 [==============================] - 1s 65ms/step - loss: 0.2991 - accuracy: 0.8672 - val_loss: 1.0204 - val_accuracy: 0.6000
Epoch 20/20
20/20 [==============================] - 1s 65ms/step - loss: 0.3676 - accuracy: 0.8328 - val_loss: 0.7683 - val_accuracy: 0.6687
```

Figure 7: Statistics were automatically printed out every time the model finishes one epoch. Importantly, the accuracy of the model was also printed out.

After this, a final batch of testing data was brought up and divided into numerous batches. These batches were then thrown at the algorithm, which had to figure out what categories the data in the batches fall into. After testing of all the batches was complete, the final accuracies of all of the batches were averaged out and given to the researcher. This final amount would show how accurate the model has become after consistent training, and will serve as a data point.

Data Collection

Data collection was extremely simple with these algorithms. The only data that needed to be collected was the final accuracy at one trial. The algorithm being tested with pure image data in its pictures was tested 12 times, and the algorithm using audio data in its pictures was also tested 12 times. This added up to 24 total trials and 24 data points to analyze.

Results

Data collection was extremely simple with the two algorithms being tested. The only data that needed to be collected was the final accuracy measured at the end of a given trial. The algorithm being tested with pure image data in its pictures was tested 12 times, and the algorithm using audio data in its pictures was also tested 12 times. This added up to 24 total trials, or 24 data points in the raw data table, which would later be analyzed through a summative data table, a calculated data table, and a statistical data table, each depicting how varied the final accuracies were.

Working Data

Final Accuracies of Image-Trained and Audio-Trained Algorithms		
Trial #	Image Data (Control)	Audio Data
1	78.12%	98.12%
2	91.75%	97.87%
3	89.88%	98.62%
4	93.75%	97.25%
5	92.50%	98.12%
6	84.75%	92.87%
7	88.25%	97.75%
8	86.25%	98.75%
9	82.00%	98.62%
10	91.87%	98.50%
11	90.62%	89.00%
12	83.38%	94.25%

Table 1: Raw data table, listing accuracies of each trial. The highest possible accuracy is denoted by 100%.

Summative Data Table for the Final Accuracies of Image-Trained and Audio-Trained Algorithms		
	Image Data	Audio Data
Mean	87.76%	96.64%
St Dev	4.87%	3.04%
Var	0.24%	0.09%
N	12	12

Table 2: Summative data table.

Absolute Deviation of the Final Accuracies of Image-Trained and Audio-Trained Algorithms		
Trial #	Image Data (Control)	Audio Data
1	-9.64%	1.48%
2	3.99%	1.23%
3	2.12%	1.98%
4	5.99%	0.61%
5	4.74%	1.48%
6	-3.01%	-3.77%
7	0.49%	1.11%
8	-1.51%	2.11%
9	-5.76%	1.98%
10	4.11%	1.86%
11	2.86%	-7.64%
12	-4.38%	-2.39%

Table 3: Calculated data table. This table provided every accuracy's distance from the mean accuracy of its group, or its absolute deviation.

Groups T-Test Used On	P-value
Image and Audio Data	0.000234283603

Table 4: Statistical data table. The t-test used to find the p-value in this study was a 2-tailed paired t-test. The test was 2-tailed because the hypothesis was that there would be a significant difference between the accuracies of an algorithm using image data and an algorithm using audio data, but did not specify as to whether image data or audio data accuracy would be higher. At the same time, the test was a paired test because, similar to a person being tested, the algorithm being tested was the same with every trial, but with a different dataset based on what group the trial belonged to.

Presentation of Results

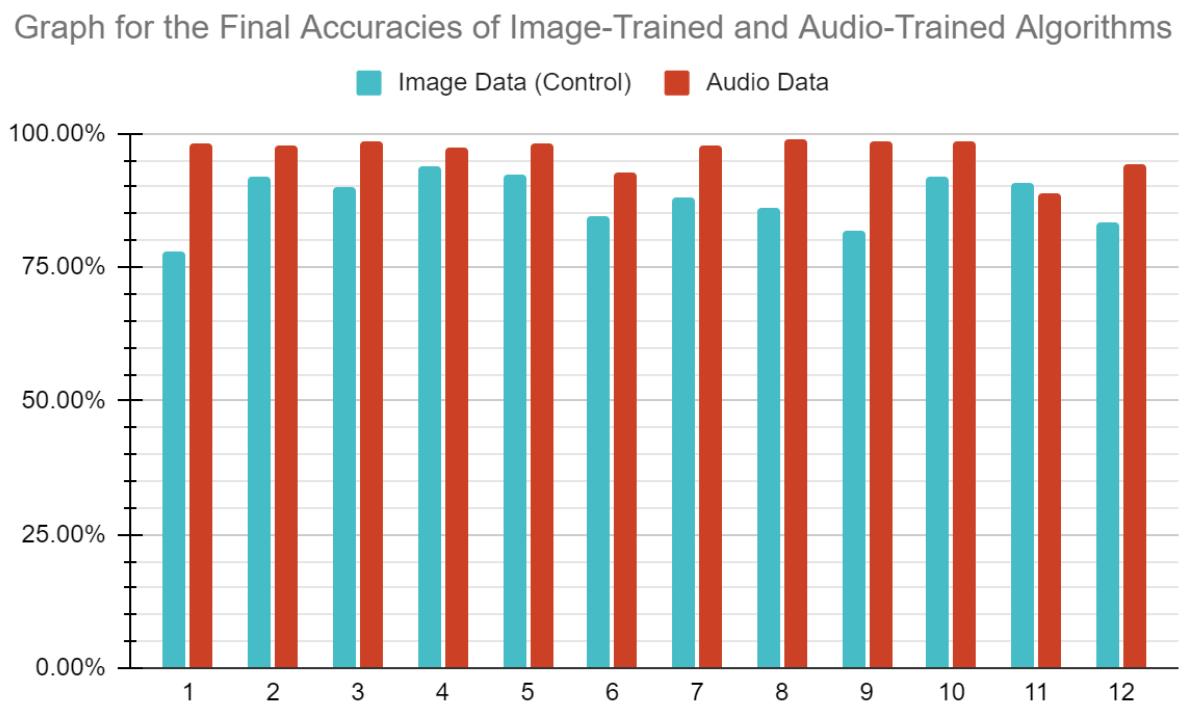

Figure 8: Here, the accuracies achieved by the classification algorithm were visualized. The columns of the chart are paired together based on what trial the results were recorded. The numbers underneath those columns represent the trial number.

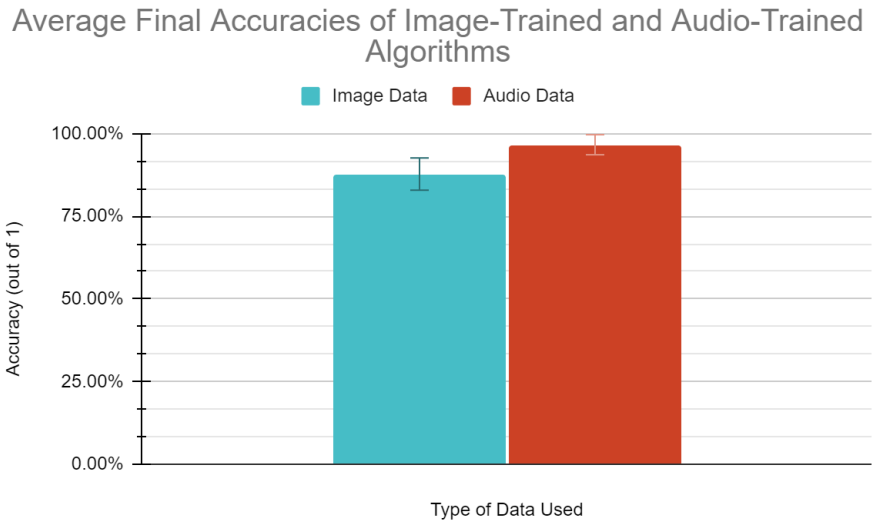

Figure 9: This chart uses a measure of center to show the mean of the accuracies of each group, or the value that the accuracies are clustered around. Meanwhile, the error bars represent one standard deviation of each of the groups, with 68% of the data being guaranteed to fall within this boundary due to the properties of standard deviation.

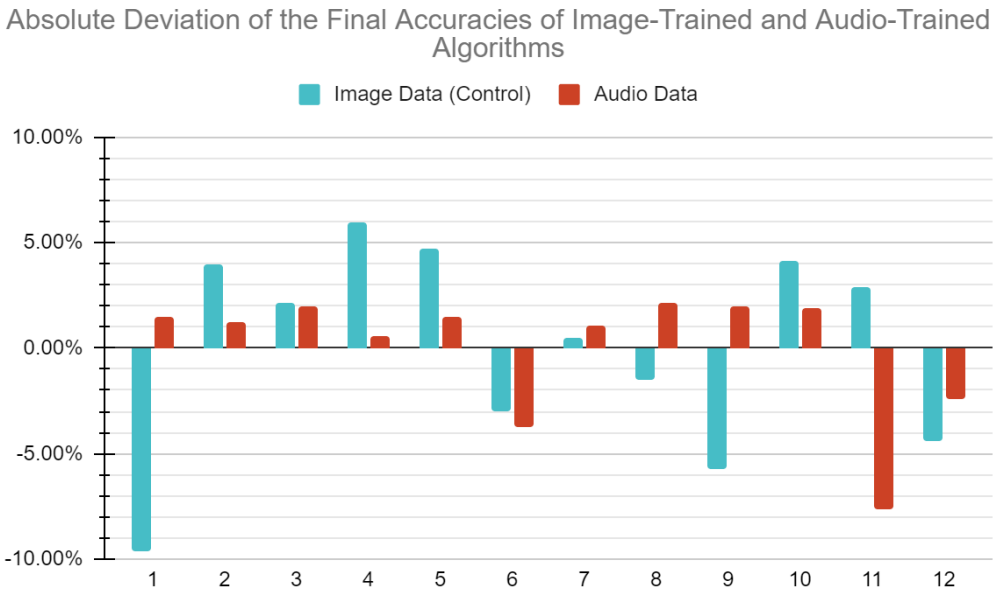

Figure 10: In this chart, the absolute deviations of the accuracies are plotted and visualized between a range of ±10.00%. As previously, the numbers at the bottom of the graph represent the trial during which these results were recorded.

Discussion:

Summary of Results and Conclusions

Some conclusions can be drawn regarding the final results of the study. Namely, the claim made prior to the commencing of the study was that: *there is a significant difference between the accuracies of image data and audio data when used in a classification algorithm.* The opposite of this, the null hypothesis, stated that: there is not a significant difference between the accuracies of image data and audio data when used in a classification algorithm.

In short, the final p-value ended up being 0.000234283603, which is well below the usual value of 0.05 for alpha. Since p-values < 0.05 tend to signify significant differences between two groups – Reject the null hypothesis, *and support the alternative hypothesis.* The data do support that the accuracies of algorithms using image data are significantly different from algorithms using audio data.

This fulfills the prediction made earlier regarding the idea that the accuracies end up very different from each other by way of the properties of the data itself. However, this conclusion is not fully substantiated due to some of the observations made while analyzing the data. The first observation regarding this is that the accuracy of audio data is unusually high for a quickly-trained algorithm with not as many layers as many professional algorithms. Given that the accuracies generally always hit at least 97.00%, the algorithm was able to perform extremely well, despite not having gone through that many layers or epochs to perfect its code.

Another observation that affected the conclusions of this study was that the standard and absolute deviations were fairly low for both the image data and for the audio data. 68% of all of the data in the audio tests fall within one standard deviation of the mean. With the mean being 96.64%, and the standard deviation being 3.04%, the chances of the accuracy ever dipping below 90% was very rare, and the upper bound of the 68% limit was 99.68%, which was an almost-perfect score. These observations mean that, while the results did happen to be significantly different in this study, this cannot be known to apply to all cases regarding audio and visual data.

Some unforeseen variables could have played a role in this study's improbable audio data results. The first of these is that the size of the datasets used in this study were fairly small. This was due to the fact that it was very difficult to find high-quality audio of dogs online. While all the data came from reputable sources, these sources were few and far between, which meant that the study could only be carried out with 500 total audio files of dogs. Since the amount of files for each training and testing group had to be the same, there could only be 1000 images in either the image or the audio datasets. In ML, this is akin to having a small sample size for data. This is because there may not have been enough data for the model to have found valuable or common traits among more pictures or sounds of dogs, and some of the traits found in the dogs in these pictures could be nonexistent for pictures of dogs that were not in the original dataset.

Another important limitation was that some data involved in the study, especially with dog audio, may have been duplicated, which caused more copies of it to exist within the study. If the same audio occurred multiple times within training data, the data started to weigh the traits of that audio much more than the traits of other pieces of audio, which meant that the algorithm would get very good at identifying that piece of data as a cat or a dog, depending on what label it had in the beginning. This may have happened over the course of the experiment, as Marc Moreaux's Kaggle dataset seemed to have some repeats in training and testing data. The accuracy could have been artificially inflated due to this.

Future Studies and Research Impact

While the original claim made has been proven to be true, this study has many limitations with regards to the data that could be found online and used to train the algorithms. Many new questions have been raised in regards to what conditions may affect the accuracy of the final model. For one, it is not known how many repeats in data may affect the accuracy of a model when testing and training, especially since the extent to which the repeats in data had an adverse effect on accuracy is not known. This is an important issue to consider because some amount of repetition in the data of

the datasets could ruin the results of an entire experiment. A future research study could tackle the problem of how different amounts of data repetition can affect the final accuracy of training and testing data.

Future studies can also tackle the best methods to acquire data for datasets, since this process took the most time out of any of the work done in this study. There are already a few methods that were considered over the course of this study. For example, a Python library called bing-image-downloader was almost used to scrape and download images based on keywords that one could type into the Bing search engine. If this were to be used in image classification by making datasets consisting of the results for cats and dogs on Bing, the results could be vastly different from the results obtained in this study.

Once more steps are taken to make image and audio data analysis more accurate and more accessible, work can be done with audio data to give it some of the abilities of image data in machine learning contexts. Studies have been working on and researching the ability of classification algorithms to take vague, low-level details in a picture and classify them through high level means (Vailaya et al.). For example, while a picture could be classified as having been taken indoors or outdoors, scientists are hoping to give classification algorithms the ability to determine whether a picture outdoors is taken in a city or a rural area based on details that would otherwise be blurry or imperceptible. This work can hopefully be translated to audio analysis as well, in order to analyze more complex sounds such as speech when distorted or unclear.

If this study were to be redone in the future, its primary focus should be on making comparisons between image and audio data by finding subjects that have more comparable data and can work with larger datasets. This will make any new accuracy results more precise, and will also be able to prove or disprove the hypothesis that was made at the beginning of this study.

Acknowledgements:

I would like to acknowledge the following groups for their valued contributions to this study. Without these people, this study may not have been possible. First off, I acknowledge the efforts of the Monmouth County Board of Commissioners, for their support and funding of the MCVSD. The MCVSD Board of Education and Administration, including Dr. Charles Ford and Mr. Sean Meehan, has also been instrumental in my study through the work that it does for students and expanding their opportunities by encouraging programs like research at this school. I am grateful to the HTHS Faculty and Administrators for the work that they do to keep our school running and teaching the minds of the next generation. In particular, thanks go out to Mr. Kevin Bals and Mr. Tim McCorkell, the principals of High Tech from 2021-2022, who continue to support this school's research program. In addition, this study would not have been done without the support of Dr. Ellsworth, who helped me formulate this research question over the summer. Finally, I appreciate the efforts of Daniel Zlotnick and Anish Pallati, who helped with clearing up concepts regarding CNNs and spectrograms.

Bibliography:

Abadi, Martín, et al.. “TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems.” arXiv, 2016., arXiv:1603.04467

- This source told me about how TensorFlow works, as well as some of its history, especially put into context with the history of CNN-related algorithms such as those working with natural language processing. I used this information in my materials and methods section.

Ajmera, Pavan K. and Harsh Sinha. “Interweaving Convolutions: An application to Audio Classification.” KDD, 2018, doi:10.475/123_4

- This source told me about CNNs and how they’ve been used in image and audio classification, along with a list of their applications in the real world. I used this information in my introduction, when talking about the historical background of CNNs.

Blum, Avrim, and Tom Mitchell. “Combining Labeled and Unlabeled Data with Co-Training.” Proceedings of the Eleventh Annual Conference on Computational Learning Theory - COLT' 98, 1998, doi:10.1145/279943.279962.

- This source provides historical context for the differences between labeled and unlabeled data. In particular, it shows how labeled data was much harder to come by due to the amount of time and work it took to produce. I used this when talking about the differences between labeled and unlabeled data.

Carpita, Maurizio, et al.. “Exploring and Modelling Team Performances of the Kaggle European Soccer Database.” Statistical Modelling, vol. 19, no. 1, 2019, pp. 74–101., doi:10.1177/1471082x18810971.

- This source told me about Kaggle's utility in the machine learning space, being both a place to create notebooks and a place to store and access datasets when needed. I used this source in the materials and methods section, when describing Kaggle as one of the tools needed for the materials.

Chakraborty, Sinjan, et al.. "Feature Map Reduction in CNN for Handwritten Digit Recognition." Advances in Intelligent Systems and Computing Recent Developments in Machine Learning and Data Analytics, 2018, pp. 143–148., doi:10.1007/978-981-13-1280-9_14.

- This source offered some criticisms of CNNs and how their new designs for feature maps would make the image classification process much faster. I used this in my introduction when talking about how CNNs have been improved over time.

Fu, Szu-Wei, et al.. "Complex Spectrogram Enhancement by Convolutional Neural Network with Multi-Metrics Learning." 2017 IEEE 27th International Workshop on Machine Learning for Signal Processing (MLSP), 2017, doi:10.1109/mlsp.2017.8168119.

- This source told me about how spectrograms are used in relation to classification algorithms, and why converting audio files into a quantifiable set of images makes it much easier to use CNNs on the audio. I used this in my introduction when mentioning how spectrograms function in an ML context.

Mohanty, Sharada P., et al.. "Using Deep Learning for Image-Based Plant Disease Detection." Frontiers in Plant Science, vol. 7, 2016, doi:10.3389/fpls.2016.01419.

- This source told me about how image classification algorithms are already proven to help with problems like identifying diseased plants, as well as the idea

that these classification algorithms have been used with living organisms before, setting a precedent for a study like mine. I used this source in my introduction, when referring to examples of CNNs being used to solve real-world problems.

Sarker, Iqbal H. “Machine Learning: Algorithms, Real-World Applications and Research Directions.” *SN Computer Science*, vol. 2, no. 3, 2021, doi:10.1007/s42979-021-00592-x.

- This source helped me understand the difference between structured and unstructured data, and why it is important to have one or the other in a machine learning algorithm. I explained these concepts in my introduction as well.

Silva, Diego F., et al.. “Applying Machine Learning and Audio Analysis Techniques to Insect Recognition in Intelligent Traps.” 2013 12th International Conference on Machine Learning and Applications, 2013, doi:10.1109/icmla.2013.24.

- In a similar vein to the source about diseased plants, this source also told me about the problem of insects acting as pests, and its solution of using machine learning to identify when an insect is nearby, at which point it can be hit with a laser. I used this article to describe more examples of real-world problems being solved by CNNs.

Vailaya, A., et al.. “Image Classification for Content-Based Indexing.” IEEE Transactions on Image Processing, vol. 10, no. 1, 2001, pp. 117–130., doi:10.1109/83.892448.

- This source told me about a problem that classification algorithms had been facing in the past, with the algorithms trying to identify obscured details from a larger image. I used this in my future studies section to talk about the direction

that audio machine learning algorithms could go into following the results of my study.