Training Artificial Intelligence Using Synthetic Data and a Convolutional Neural Network for Sound Classification of a Heartbeat

Dr. Jonathan Adams, Astrid Daugherty – College of Communications and Information at Florida State University

GOAL

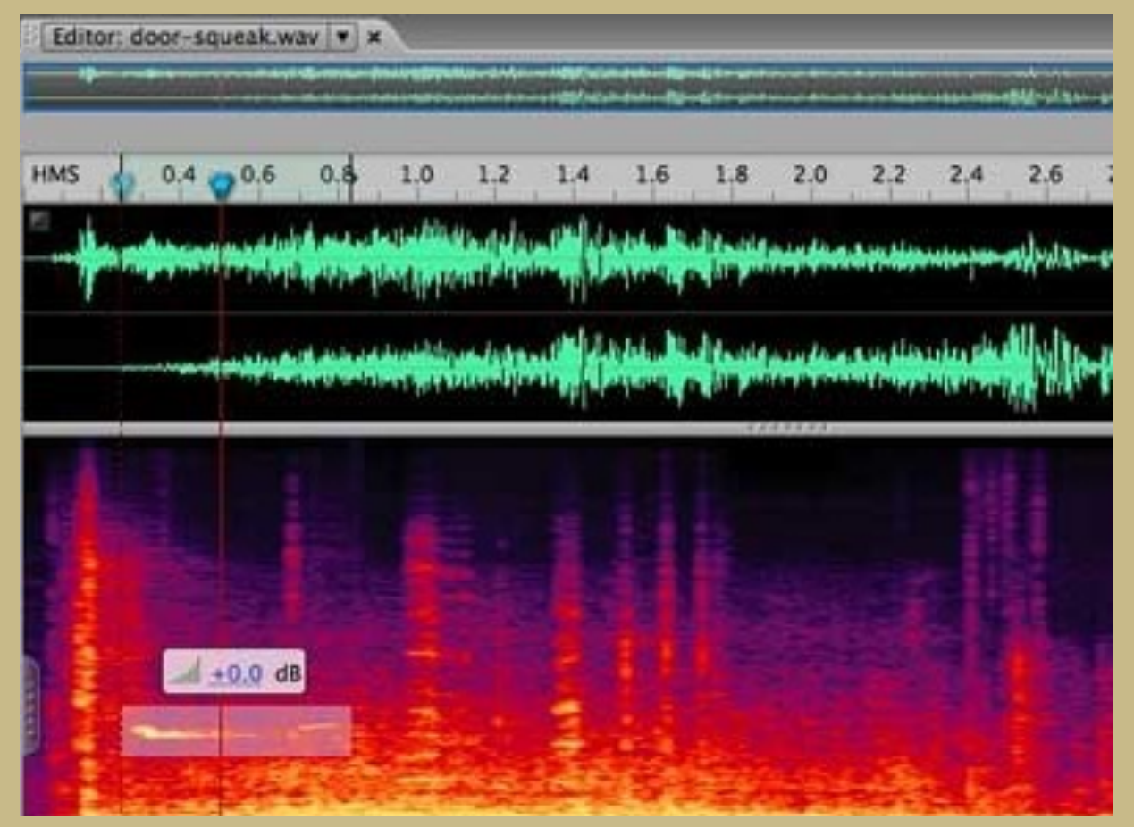
The purpose of this project is to train A.I (artificial intelligence) using synthetic data so that it's able to evaluate the sound of a human heartbeat and enhance the given data to identify abnormalities. The model used to satisfy the main goal was a CNN (convolutional neural network).

INTRODUCTION

The diagnosis of a heart disease is not as easy as hearing a heart through a stethoscope. Sometimes it requires the thorough study of a PCG (phonocardiograms), or ECG (electrocardiogram), which, at times, might not be as efficient and straightforward as one wishes it should. Doctors train their whole lives to be able to identify and treat the emergence of one. Technology now aims to help hardworking doctors do the hard work for them, so that the diagnosis is efficient and accurate. With this project we aim to improve the efficiency of the training of the A.I. by using synthetic data.

METHODS 1

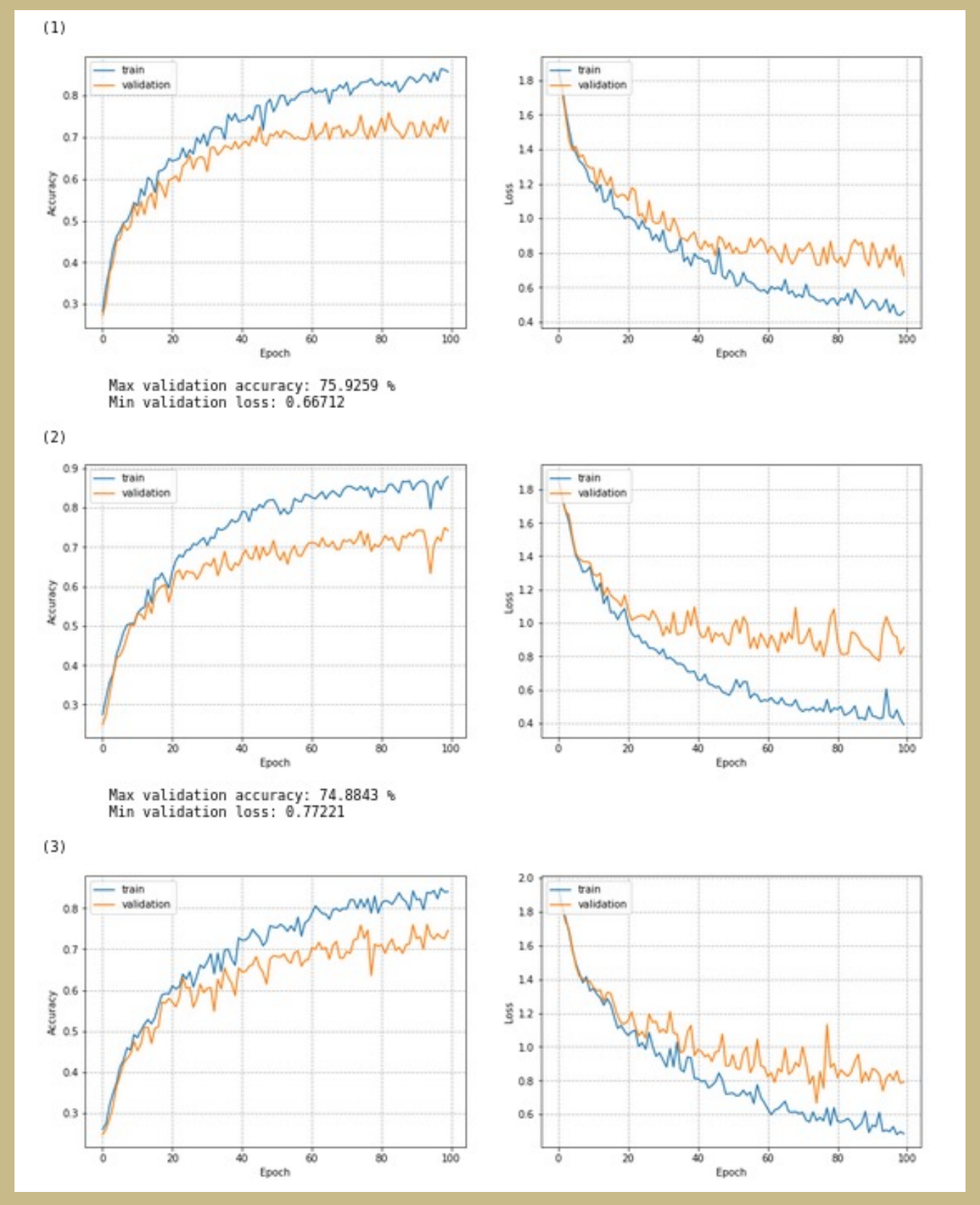
Different things were being done at the same time. We worked in a literature review concerning similar and different methods of training A.I. for classification purposes. This gave a foundation for the set up of the algorithm, and the type of data that the related works were able to obtain. Most of the articles selected for review showed that the most popular neural network for classification is the CNN, but we selected a high percentage of works which used different algorithms, like RNNs, LSTM, or Fuzzy NN as part of the training process. None of them showed the use of synthetic data for the training of the algorithm.



Instead of studying amplitude over time, the CNN studies the spectrographs of audio as frequency over time so that it can be compared and classified.

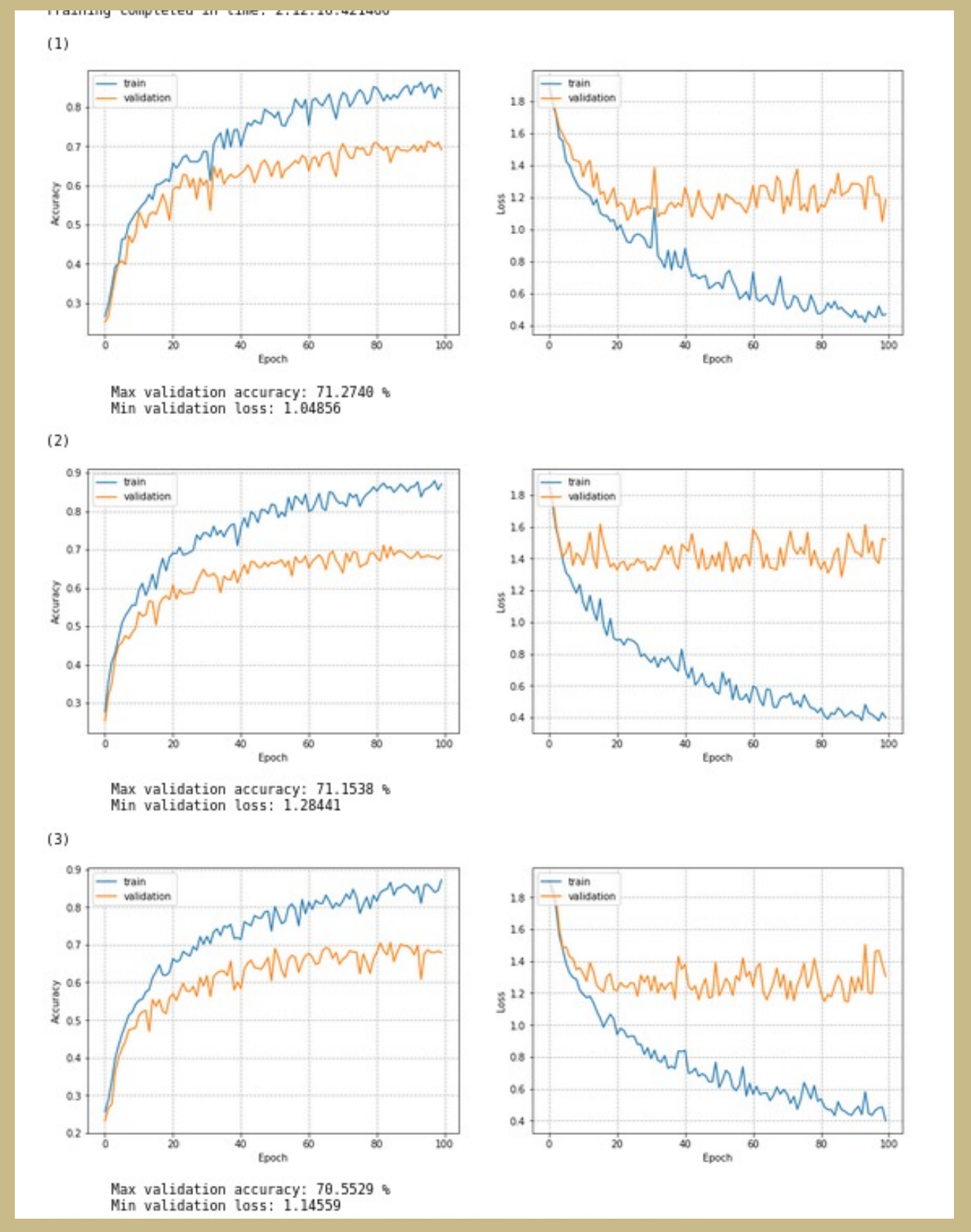
ACCURACY

After various rounds of training, the CNN we used was able to achieve an accuracy of 77%, which can be improved by spending more hours of training with more data sets. Data collected concerning the accuracy of the algorithm is shown below.



```
Max validation accuracy: 75.9259 %
Min validation loss: 0.66712
```

```
Max validation accuracy: 74.8843 %
Min validation loss: 0.77221
```

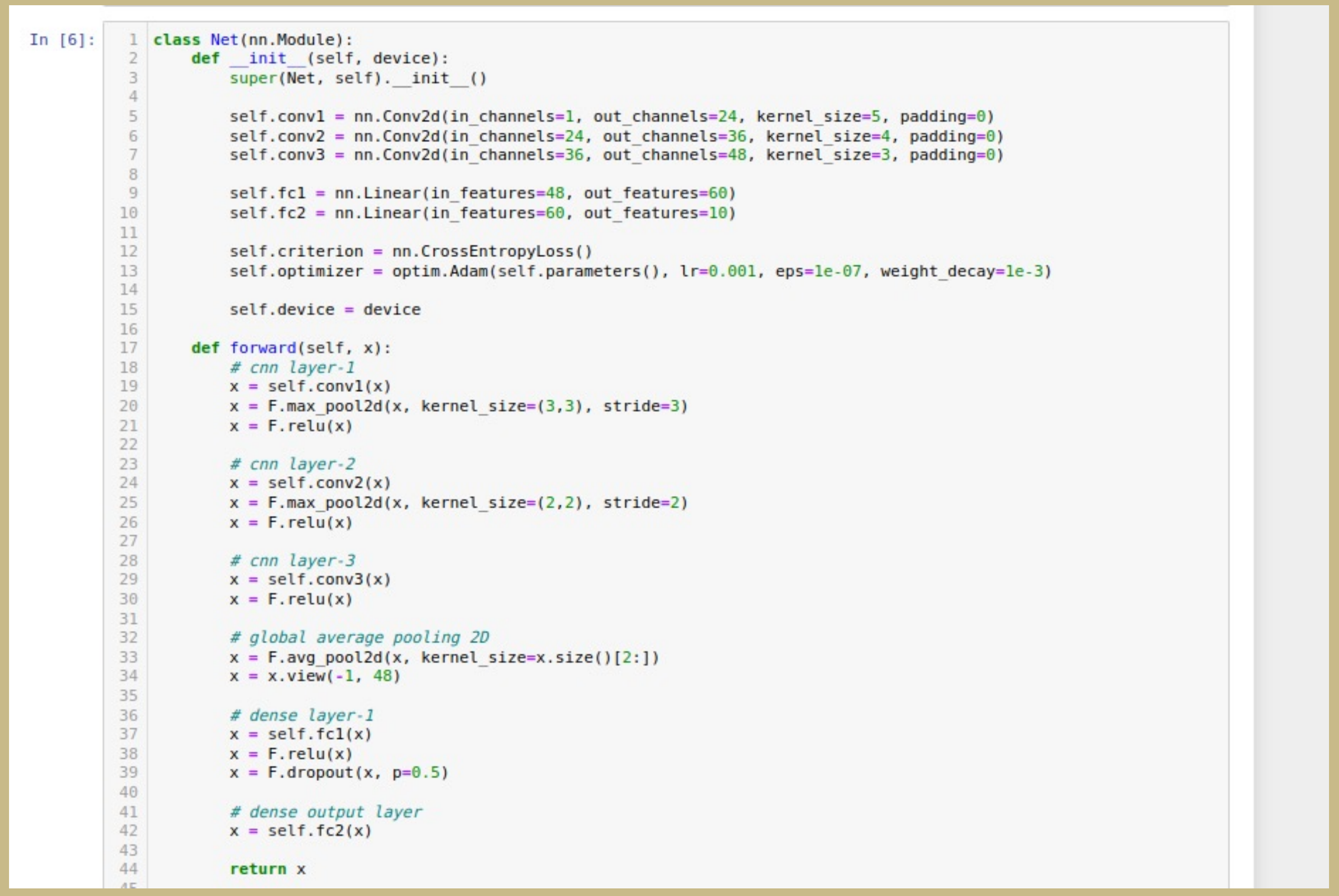


```
Max validation accuracy: 71.2740 %
Min validation loss: 1.04856
```

```
Max validation accuracy: 71.1538 %
Min validation loss: 1.28441
```

```
Max validation accuracy: 70.5529 %
Min validation loss: 1.14559
```

The graphs above depict the increasing accuracy of the algorithm for every round of training on environmental sounds.



```
In [6]:
class Net(nn.Module):
    def __init__(self, device):
        super(Net, self).__init__()

        self.conv1 = nn.Conv2d(in_channels=1, out_channels=24, kernel_size=5, padding=0)
        self.conv2 = nn.Conv2d(in_channels=24, out_channels=36, kernel_size=4, padding=0)
        self.conv3 = nn.Conv2d(in_channels=36, out_channels=48, kernel_size=3, padding=0)

        self.fc1 = nn.Linear(in_features=48, out_features=60)
        self.fc2 = nn.Linear(in_features=60, out_features=10)

        self.criterion = nn.CrossEntropyLoss()
        self.optimizer = optim.Adam(self.parameters(), lr=0.001, eps=1e-07, weight_decay=1e-3)

        self.device = device

    def forward(self, x):
        # cnn layer-1
        x = self.conv1(x)
        x = F.max_pool2d(x, kernel_size=(3,3), stride=3)
        x = F.relu(x)

        # cnn layer-2
        x = self.conv2(x)
        x = F.max_pool2d(x, kernel_size=(2,2), stride=2)
        x = F.relu(x)

        # cnn layer-3
        x = self.conv3(x)
        x = F.relu(x)

        # global average pooling 2D
        x = F.avg_pool2d(x, kernel_size=x.size()[2:])
        x = x.view(-1, 48)

        # dense layer-1
        x = self.fc1(x)
        x = F.relu(x)
        x = F.dropout(x, p=0.5)

        # dense output layer
        x = self.fc2(x)

        return x
```

Code of the convolutional neural network (CNN)

METHODS 2

For the expansion of data, we started with 23 audio samples, where each represented a class/type. These files were transformed into over 2000 samples. The filter feature had 8 types: afd_mod and sub, fat_mod and sub, jaw_mod and sub, and sat_mod and sub. The tempo feature had four types, 0.5, 0.75, 1.25, and 1.5. The pitch feature also had types -5 and +5. This allowed us to make various combinations to create synthetic data used to compare it to the original or "normal" sound. The benefit of this is to use a small number of quality recordings and generate a lot of sets from them. In the meantime, we also trained the CNN to detect other sounds unrelated to heartbeats so that the algorithm knows what is looking for.

SYNTHETIC DATA

We were able to convert 23 audio files obtained from the public data set UrbanSound8k into over 2000 audio files by applying different components: filters, change of pitch, and tempo. This process is called "augmentation".

CONCLUSION

We have achieved to train a CNN using synthetic data. This makes the training process more efficient by being able to generate large amounts of data crucial to the machine learning of the algorithm, without relying on large amounts of individual recordings.

REFERENCES

Rath, A. (2021, August 31). *An exhaustive review of machine and deep learning based diagnosis of heart diseases*. SpringerLink. https://link.springer.com/article/10.1007/s11042-021- 11259-3?error=cookies_not_supported&code=4c0786b4- 8348-4855-9911-753ae2c3a45f#citeas

Chen, W., Sun, Q., Chen, X., Xie, G., Wu, H., & Xu, C. (2021, May 26). *Deep learning methods for heart sounds classification: A systematic review*. MDPI. Retrieved March 9, 2022, from https://doi.org/10.3390/e23060667