USING 1D CONVOLUTIONAL NEURAL NETWORK (CNN) TO CLASSIFY AUDIO FREQUENCIES

Chad Kakau

Victoria University, Wellington

ABSTRACT

This paper develops a simple 1D convolutional neural network to learn and classify audio signals. The paper briefly summarises the 1D convolutional neural network, in particular the effects of pool sizes, and stride length on CNN performance. Implementing a simple 1D CNN results in better validation and test results for smaller pool sizes and shorter stride lengths. Implementation also includes reviewing a mixture of frequencies (summation of two fundamental frequencies), that results in comparable performance to the single frequency models at highest stride length and pool sizes. The paper concludes by acknowledging the relative simplicity (i.e. repepetitive and constant wave form) of the reference frequencies leads to good performance of the CNN, before considering testing the model on more complex signals in the future.

***Index Terms*—** 1D Convolutional neural network, 1D CNN

1. INTRODUCTION

Neural networks continue to provide state of the art results for image and audio recognition applications [1] including classification. 1D CNN uses input data that can be traversed in a single direction, so audio files make excellent candidates for machine learning models [2]. By combining with a range of other neural network methods, we can implement quite complex models for any range of tasks, in this case for classification of noisy audio signals into one of five frequencies. This implementation will generate five reference signals as 2 second tonal frequencies, then add noise to the data before applying a 1D CNN, a pooling layer, flattening the data and then creating two dense layers, the first with 64 nodes, the second and final layer, with five nodes, each representing one of the five reference frequencies.

2. AUDIO SIGNALS AND 1D CNNS

For this implementation we begin by generating reference frequencies for use as the class or label items. Then we will put those frequencies through a 1D CNN, adjusting stride length and pool size to tune the model's performance.

2.1. Audio and Hamming window

At its most basic, an audio signal can be represented as an array of numeric values, each indicating a quantity of energy received (or sent) at a specific time [3]. Humans can perceive audio frequencies between $15 - 20000$Hz [4], with that perception deteriorating with age, especially for higher frequencies.

Because we will feed the audio signals into a neural network, we need to ensure that each signal has the same shape (i.e. number of time-steps or sample points, and the same number of channels) [3]. We can achieve this using a window function, in this case we will use a Hamming window, which is represented as a known length (of time-steps), tapering cosine curve with tails > 1 [5] [6]. By combining a Hamming window with a frequency we can generate an output array of a known length, that can be converted to a soundfile [7]. To generate the output frequency we multiply our Hamming wndow by $\sin(\omega t)$, where $\omega = 2\pi f$, where f is the input frequency. ω is also known as the "angular frequency".

We take the following frequencies:
$f \in 317Hz, 780, 1234Hz, 2017Hz, 3106Hz$

2.2. Stride length and pooling

In this 1D CNN we will tune to key parameters: stride length and pool size. The 1D CNN passes a filter across the input array, the filter performs some function against each element in its area and outputs the result of that function to the next layer [2]. The filter may pass one element at a time (so a 3-bit filter will cover each element three times), giving a stride length of 1, or it may pass across the element faster by taking longer strides. This increases the speed at which the filter processes the array but also reduces the amount of information available to the next and subsequent layers in the model.

Pooling is a technique where a group of elements are represented by some feature of that group, usually the max value or the average. Max pooling allows for the detection of a feature, but may not provide detail as to the location of that filter within the pool. Again, pooling reduces the amount of infor-

mation available to subsequent layers, but brings an increase in speed. We should expect performance to degrade as the stride length and pool size increases [2].

For the purpose of comparison, we present these five frequencies in two ways: the upper plots show amplitude (y axis) over time (x axis) and the plots all have the same time scale. The lower plots show the fast fourier transformation of each frequency, this time with frequency along the x axis, showing a clear spike at the relevant frequency. Using fast fourier transformation allows us to decompose a complex signal into its component parts and can be useful for identifying the contributing frequencies in order to isolate and amplify or filter that frequency [7].

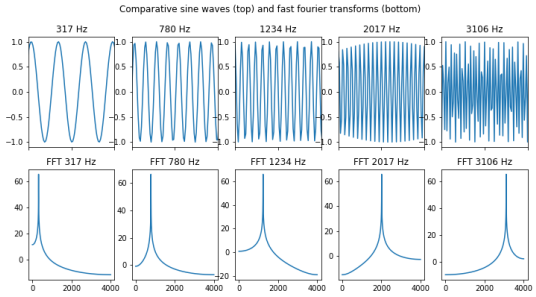


Fig. 1. "Initial frequencies and fast fourier transforms"

(Fig 1) shows the first five frequencies and should be compared against the later plot showing mixed frequencies (Fig 2) :

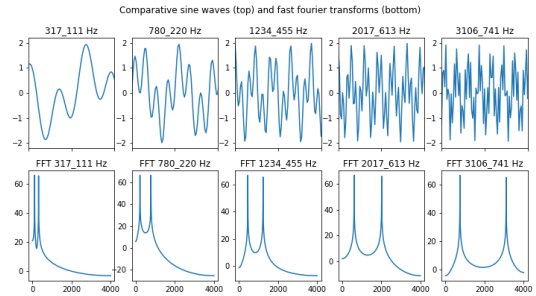
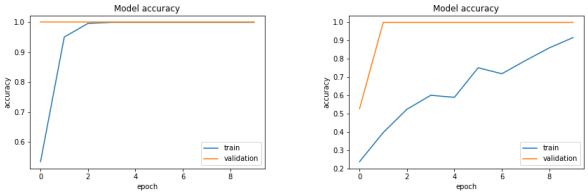


Fig. 2. "Mixed frequencies and fast fourier transforms"

3. RESULTS

The results show that the 1D CNN can learn to classify audio signals with very high accuracy, including when using quite high stride length and pool size.

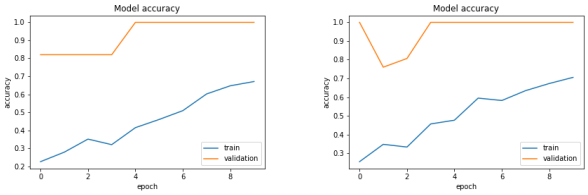
We compare performance of 1D CNNs, with strides of the same length (8) and pool = 2, then pool 8 (Fig 3). We can see that performance with a smaller pool size (i.e. pooling fewer elements) is better than with a larger pool. The training



(a) Pool = 2, stride = 100 (b) Pool = 8, stride = 100

Fig. 3. Performance of 1D CNN with different pool size

accuracy improves at a slower rate for pool = 8 and never reaches the full accuracy in the model. Validation and test results for both models reach 100% early.



(c) Pool = 8, stride = 180 (d) Pool = 8, stride = 180

Fig. 4. Performance with strides of 180 and mixed frequencies

We can also compare the results of models with pool = 8, stride = 180 but using the baseline frequencies versus mixed frequencies. (Fig 4) shows very similar performance for both models, both reaching maximum training accuracy of around 60% after 10 epochs of training, but with validation accuracy at 100% by the fourth epoch and test accuracy of 100%.

4. CONCLUSION

1D CNN performance on predicting 1-channel audio tones appears to be dependent on stride length and pool size, and performs comparably when comparing either a single sinusoidal tone or a complex combination of two sinusoid waves. The effect of stride length seems fairly intuitive, since the length of each stride determines either how many times an element is sampled by each filter on each pass (i.e. low stride length) or how many individual elements may be missed altogether (i.e. high stride length) by each filter on each pass. For an array of the nature and size in this implementation, the stride length can become quite large before training accuracy is noticably degraded.

The effect of increased pool size is more readibly noticed when downstream from increased stride length because as information is lost (i.e. the filter strides past elements) there are fewer elements to pool and key feature elements (e.g. peaks and troughs) of the underlying array may be lost or missed, making learning more challenging. The trade-off for loss of

information is reduced computation (due to fewer elements), resulting in faster models.

It would be interesting to test a model against more complex or erratic audio signals (e.g. ambient noise on the street to detect vehicle types, or birdsong to classify birds in a neighbourhood etc.), similar to the models generated at [1] and [8]. A 1D CNN in these situations should provide very useful results, although tuning would likely result in degraded performance at lower stride and pool values, which should in turn bring longer processing times.

Link to colab site for code:
`https://colab.research.google.com/drive/13lFG3ZxG-QAvMwT4B3hzLTMdNQNgMAkG?usp=sharing`

5. REFERENCES

[1] Jason Brownlee, "1d convolutional neural network models for human activity recognition," sep 2018.

[2] "Convolutional neural network (cnn) tensorflow core," .

[3] "Audio data preparation and augmentation tensorflow i/o," .

[4] Sanjeev Kulkarni, "Lecture notes for ele201 introduction to electrical signals and systems," 2001.

[5] "On the use of windows in digital signal processing," .

[6] Frederic J. Harris, "Multirate fir filters for interpolating and desampling," in *Handbook of Digital Signal Processing*, pp. 173–287. Elsevier, 1987.

[7] "05-frequency-analysis," .

[8] "Simple audio recognition: Recognizing keywords tensorflow core," .