John Ottenlips

Corey Sery

December 2, 2016

Machine Learning Drums with Neural Networks

For our project, we created a program that used supervised learning methods to learn drum samples and make accurate predictions on the drum type. To accomplish this, we used the Tensorflow library to create a couple of different networks. We tried a multilayer perceptron network and a convolutional highway network.

For the data set, we collected kick, snare, tom, and hihat samples from multiple libraries. First, we down sampled the tracks to 11.025kHz, or one fourth of the original quality. This allowed us to run the process faster, the cost was loss of higher audio frequencies. To format the data, we created a 4d tensor, or multidimensional array, of the audio. The dimensions were comprised of tracks, batches (to represent chunks of time), frequencies, and complex amplitude (representing amplitude and phase). To prevent over fitting our data, we divided our data into three randomized sets: testing, training and validation. The training data was what Tensorflow used to train the network. Tensorflow would use the testing set along the way to verify the training data was not being overfitted. But this could possibly lead to the testing data being overfit as well. For this, we used a validation set that was completely independent of the training and the model. This was the best way to be sure that the model has generalized and will work with unseen data.

Neural networks are composed of different types of hidden layers. The hidden layers we used on the multilayer perceptron network were batch normalization, softmax, and elu layers.

The batch normalization layer helps optimize the relationship between performance and learning rate. In traditional neural networks a higher learning rate leads to problems like vanishing or exploding gradients during back propagation. Batch normalization allows the neural net to make use of a higher learning rate without these side effects. Normalizing layer inputs is important because it is how we address the problem of covariate shift, when distribution for each layers input changes due to changing parameters of previous layers. Batch normalization layers make normalization of layer inputs part of the model, making input initialization less important while also increasing the performance of the network.[6] The softmax layer is used for multidimensional classification and is usually the final layer when used for supervised classification, which is important for handling samples of many different drum classifications beyond just snare and kick. This layer transforms the output vector of values into a vector of values between 0 and 1 that add to 1.[1] The ELU, exponential linear unit, layers make the learning faster, combat the vanishing gradient problem, and lead to higher classification accuracy. Modern neural nets have moved away from sigmoid activation functions and use ELU instead because of their many benefits. The ELU also allows for negative values, giving it the ability to push the mean unit activations closer to zero, batch normalization also does this.[2] The different types of hidden layers of neural networks give the architecture flexibility and help to optimize their accuracy and efficiency.

Convolutional Neural Networks, CNNs, have a depth of layers that allows for more complex models. These networks also have pooling layers periodically in between the convolutional layers. Pooling allows for surrounding data to be grouped with the current node. This allows for the network to look at numbers in a larger context. Convolutional layers use a

filter to repeatedly go over the input to identify different features. Pooling layers reduce spatial size and help prevent overfitting. To reduce the spatial size these layers “downsample” the width and height of the convolutional layers.[3] CNNs help us make more complex models that can identify multiple features. This type of network is also the type of network used for Google’s Deep Dream project.

To optimize our stochastic gradient descent we used the Adam algorithm. In our tests Adam outperformed standard SGD, stochastic gradient descent. This may be because Adam uses adaptive learning rates for different parameters based on previous moments of the gradients, which means that they need less tuning than a traditional stochastic gradient descent.[4] Adam is also known for its ability to deal with noisy parameters, which is significant when dealing with audio samples.

For our first attempt we passed the audio tensor to two different neural net architectures. The first one was a multilayer perceptron network. It used the Adam method for stochastic optimization. Adam was successful because it handles large, noisy data very well. Audio signals contain a lot of extra noise that may not be relevant to classification. This network also utilized batch normalization, softmax and elu hidden layer. This architecture was able to correctly guess, between a kick and a snare sample, with up to 100% accuracy.

The second network we used was a convolutional highway network which also used the Adam method. The difference with convolutional networks is that they have layers of depth as well as width. For a convolutional highway network all of the hidden layers are fully connected across unimpeded “information highways” to make it more efficient when processing deep layers. This extra flexibility allows the network to train more complex models than some of it’s

more simpler counterparts. This particular network was able to correctly identify four different samples: kick, snare, tom, and hi-hat; with up to 98% accuracy.

In the future, we would like to use the model to generate audio as well as classify it. Similar to what the Google Deep Dream project does with images, by putting a feedback loop on our classification model. The Google Deep Dream is able to transfer styles and features of one image onto another image through the use of its convolutional layers. It would be interesting to see if style transfer would work with different styles of audio and different genres of music. If style transfer works, we could make an innovative audio application that would be very fun to experiment with.

Citations

Bishop, C. M. (2006). Pattern recognition. *Machine Learning*, *204*. http://users.isr.ist.utl.pt/~wurmd/Livros/school/Bishop%20-%20Pattern%20Recognition%20And%20Machine%20Learning%20-%20Springer%20%202006.pdf

This book went in depth about machine learning and artificial intelligence like probability theory, model selection, decision theory, models for regressions and more things that were outside the immediate scope of what we needed to know for this project. We used it for general background research as well as to get information about how the softmax function is used with for classification problems.

Clevert, D. A., Unterthiner, T., & Hochreiter, S. (2015). Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*.

This document was published as a conference paper at the International Conference of Learning Representations in 2016. In the paper, the authors discuss exponential linear units and the benefits of using them over rectified linear units and other sigmoid activation functions. The results of their study showed that ELU layers significantly speed up the machines ability to learn compared to ReLU with the same architecture.

Karpathy, A. (2016). Convolutional Neural Networks. http://cs231n.github.io/convolutional-networks/#add

Andrej Karpathy is a research scientist at OpenAI that specializes in Deep Learning, Generative Models, Reinforcement Learning. He is also a Ph.D. student at Stanford. In this document Karpathy walks through how convolutional networks work layer by layer. He explains the concept of convolutions and includes many helpful diagrams.

Kingma, D., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

This conference paper by Diederik Kingma and Jimmy Ba discusses the advantages of the Adam algorithm for stochastic optimization of neural networks. The paper discusses the origins of the algorithm and walks through the pseudocode. It also shows specific examples of its performance vs other algorithms like SGD and AdaGrad on a multilayer neural network. The paper clearly explains how Adam takes advantage of lower order moments of gradients to compute individual adaptive learning rates for different parameters. Adam deals well with networks involving many noisy parameters and requires very little tuning to be accurate.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097-1105).

Ilya Sutskever (Research Director at openAI), Alex Krizhevsky (Google), and Geoffrey E. Hinton (Google & University of Toronto) trained a large convolutional neural network to classify 1.2 million high resolution images. They discuss their techniques to prevent overfitting using dropouts. They also discuss the architecture of their convolutional network which consisted of convolutional, max pooling, fully connected, and softmax layers.

Loffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

The authors, Christian Szegedy, a research scientist at Google, and Sergey Loffe, a machine learning expert at google, showed how adding bath normalization layers affected the performance of the neural network. Their results found that they improved upon the best ImageNet classification, exceeding human accuracy. They also found that batch normalization helped their network achieve the same accuracy with 14 times fewer training steps.

Srivastava, R. K., Greff, K., & Schmidhuber, J. (2015). Highway networks. *arXiv preprint arXiv:1505.00387*.

Rupesh Kumar Srivastava a PhD student at the University of Lugano in Lugano, Switzerland studies artificial intelligence and machine learning. He is guided by professor Jürgen Schmidhuber. Highway networks were created to more efficiently train deep networks. The main idea is to allow data flow across multiple hidden layers. We used it because modeling audio is rather complex and a simpler linear model, would most likely be less effective.

(2016). Tensorflow Documentation. https://www.tensorflow.org/versions/r0.12/api_docs/python/index.html

This is the current documentation for the Tensorflow library. We used it to learn how Tensorflow works and to understand what parameters we were changing in our network.

(2016). Tflearn Documentation.
http://tflearn.org/

Tflearn is a higher level API to the Tensorflow library. We thought that Tensorflow would be the most effective for the task, but the learning curve was a bit high just to get off the ground. This library solved that problem for us. We modeled our network off of the example networks they give in their documentation.