TECHNICAL UNIVERSITY IN KOSICE

Stochastic modeling and data analysis

2020

Authors: Kukhar Denys,

Lutak Kristian

Stochastic modeling and data analysis

I. Scope knowledge needed

A. Intro

Nowadays, deep learning is more and more used for Music Genre Classification: particularly Convolutional Neural Networks (CNN) taking as entry a spectrogram considered as an image on which are sought different types of structure.

B. Description of CNN

Convolutional Neural Networks (CNN) are very similar to ordinary Neural Networks: they are made up of neurons that have learnable weights and biases. Each neuron receives some inputs, performs a dot product and optionally follows it with a non-linearity. The whole network still expresses a single differentiable score function: from the raw image pixels on one end to class scores at the other. And they still have a loss function (e.g. SVM/Softmax) on the last (fully-connected) layer and all the tips/tricks we developed for learning regular Neural Networks still apply.

ConvNet architectures make the explicit assumption that the inputs are images, which allows us to encode certain properties into the architecture. These then make the forward function more efficient to implement and vastly reduce the number of parameters in the network.

They are capable of detecting primary features, which are then combined by subsequent layers of the CNN architecture, resulting in the detection of higher-order complex and relevant novel features.

C. Feaature extraction

Convolution is one of the main building blocks of a CNN. The term convolution refers to the mathematical combination of two functions to produce a third function. It merges two sets of information.

In the case of a CNN, the convolution is performed on the input data with the use of a filter or kernel (these terms are used interchangeably) to then produce a feature map.

We execute a convolution by sliding the filter over the input. At every location, a matrix multiplication is performed and sums the result onto the feature map.

D. ADAM Optimizer

ADAM is an algorithm for first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments. The method is straightforward to implement, is computationally efficient, has little memory requirements, is invariant to diagonal rescaling of the gradients, and is well suited for problems that are large in terms of data and/or parameters. The method is also appropriate for non-stationary objectives and problems with very noisy and/or sparse gradients. The hyper-parameters have intuitive interpretations and typically require little tuning. We also analyze the theoretical convergence properties of the algorithm and provide a regret bound on the convergence rate that is comparable to the best-known results. Empirical results demonstrate that Adam works well in practice and compares favorably to other stochastic optimization methods.

Adam can be looked at as a combination of RMSprop and Stochastic Gradient Descent with momentum. It uses the squared gradients to scale the learning rate like RMSprop and it takes advantage of momentum by using moving average of the gradient instead of gradient itself like SGD with momentum. Let’s take a closer look at how it works. Adam is an adaptive learning rate method, which means, it computes individual learning rates for different parameters. Its name is derived from adaptive moment estimation, and the reason it’s called that is because Adam uses estimations of first and second moments of gradient to adapt the learning rate for each weight of the neural network.

II. Dataset

A. Source description

Our source for data was a Saarbruecken Voice Database. This is a collection of voice recordings from more than 2000 persons. One recording session may contain the following recordings:

- Recording of the vowels [i, a, u] produced at normal, high and low pitch
- Recordings of the vowels [i, a, u] with rising-falling pitch
- Recording of the sentence "Guten Morgen, wie geht es Ihnen?" ("Good morning, how are you?")

B. Data used

For our project we found the best format for analyzing the voice data. Used subset of available data consists of 1554 audio tracks each 2-3 seconds long. It contains 2 categories (healthy, pathological), each represented by 777 samples. The samples are all in .wav format. Each sample represents one of three (i, u, a) vowel sounds of the same lengths.

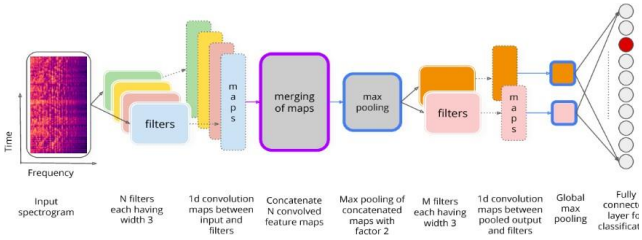
C. Demographic inforamtion of subjects

We found out that in order to distinguish the healthy and pathological voices we better use men’s half of records since their issues are more detectable comparing to women’s

ones. For easier classification of diseased person's voice record the usage of all accessible pathologies was chosen. We found all ages of subjects to be appropriate to test against as they don't affect sound much.

III. Network Design

A. Overview



B. Sctructure of CNN model used in project

Used input shape is (64, 64, 3). We used sequential model with following layers:

- Convolutional 2D layer (32 for filters, (3,3) tuple as kernel size, (2, 2) for striding)
- Average pooling 2D ((2,2) for pool size, same for striding)
- Relu activation function
- Convolutional 2D layer (64 for filters, (3,3) tuple as kernel size, 'same' for padding)
- Average pooling 2D ((2,2) for pool size, same for striding)
- Relu activation function
- Convolutional 2D layer (64 for filters, (3,3) tuple as kernel size, 'same' for padding)
- Average pooling 2D ((2,2) for pool size, same for striding)
- Relu activation function
- Flatten layer
- Dropout with 0.5 as rate
- Dense with 64 units
- Relu activation function
- Dropout with 0.5 as rate
- Dense with 2 units for output classes
- Softmax activation function

IV. Experiments

In scope of this project we have tried many strategies and applied many adjustments, both major and minor. Amongst mentioned experiments, we found out that analyzing sound performs better when pathologic is being detected based on men vowel samples, also we compared compiling/training the network using Stochastic Gradient Descent (SGD) as optimizer with Adam as optimizer. Gradient Descent works fine when we have a convex curve. But if we don't have a convex curve, which is our case, Gradient Descent fails. Hence, in Stochastic Gradient Descent, few samples are selected randomly instead of the whole data set for each iteration. Alas, we had to stick to Adam instead as our go-to optimizer. Its better sides are described below.

Adam optimizer simply led us towards better results. That's the reason we chose to stick to this. Also, we have run tests indicating of how much epochs should be optimal. After 50 epochs accuracy seems not to change much.

One thing was mentioning is that with such analyzing sound problem we basically fully rely on quality of the given data. Even if you try to examine the class of sample by ear you will definitely find yourself confused many times.

With less samples (around 500 samples for each category) we have managed to achieve 70% accuracy in predicting. After another 554 samples were loaded to the training, the accuracy began to descend. Having 777 samples for each category we have achieved accuracy of 64%.

V. Suggested Implementation and Program Flow

In order to create specgram of the sound we iteratively use specgram function from matplotlib library for conversion WAV files to PNG format. All audio files primarily placed in its directory in wav_data will have its specgram generated in corresponding folder img_data.

Next, we split 80% data for training and 20% for testing against, we perform mutations on the data such as zooming and random transformations, rescale all pixel values from 0-255, so after all our pixel values are in range (0, 1).

After this we create CNN model according to the network design diagram before compiling. Once the model is trained, we write prediction results made with our model to CSV file.