Discrimination of reflected sound signals

Master Degree
Information technology
EL MEHDI DIAB
1324316
el.diab@stud.fra-uas.de

***Abstract*—The surface of an object reflects a sound signal. Recording the reflected signal provides a time signal. This time signal results from the convolution of the incident acoustic wave with the surface properties of the reflecting object. Knowing the incident acoustic signal one can conclude on the reflecting object by analyzing the reflected signal. In this study we will discuss about using the reflected time signals to conclude on the reflecting object. As the number of objects is limited in our experimental setup, to discriminate the time signals we will apply machine learning (ML) using convolutional neural network (CNN) and label data. Furthermore, this paper also discusses our experimental result.**

Keywords—sound Signal, machine learning, convolutional neural network, time signals

I. Introduction

The sound reflection can be determined in time between the termination of the initial sound source and its reintroduction in a space.. Echo and reverberation was described by the difference between these 2 sound signals. The reflection is considered an echo if the distance is longer than one second. The reflection called reverberation when the distance is less than a second. Our human ear can't distinguish between less than one second of sound signals. Sound reflects the part of the original sound wave that is within your home. When the reflection is offset by less than a second from the initial sound signal, the human ear can hear the sound as a long signal called a reverberation. Discrimination against time signals is generally done using machine learning technology, in which algorithms are initially educated on manually annotated data and then can be used to interpret previously unknown audio data.

In this paper is organized as follows. A starting with Cleaning and Merging Data in section 2, followed by section 3 with detailed Reading time signals. Section 4 Creating a binary classifier (discriminator) by machine learning using convolutional neural network (CNN) and label data. In section 5 we will implement the experiment in a graphical user interface (GUI). Finally, experimental result and analysis is presented as section 6.

II. Cleaning and merging data

A. *Preaparing the mesuring Data*

Measuring signals was given as 2 objects namely object#1 and object#2. The first one contains 8 files; the second include 11 files. Inside of each file there is 30 csv-file that measured time signal. One time signal consists of 16384 scan point. As it is mentioned on table below:

Table 1 Measuring Test Data

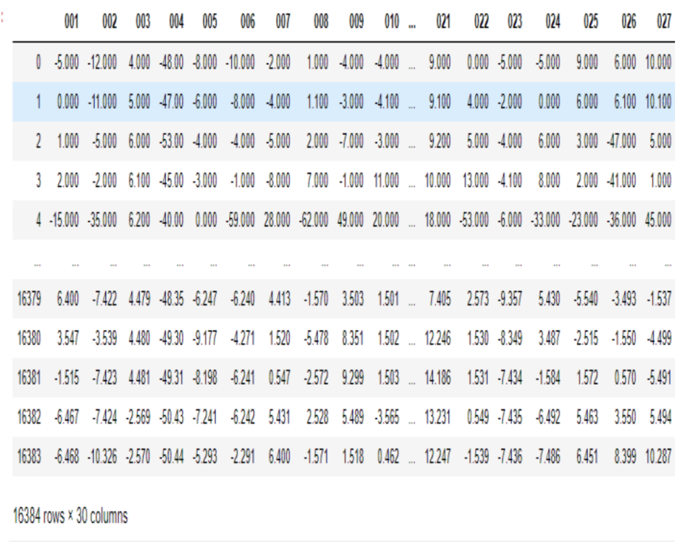
Object#1 files	Object#2 files	csv-file	Scan Point
side 1-0.5	front-black 0.5	30	16384
side 1-0.8	front-black 0.8	30	16384
side 1-1.2	front-black 1	30	16384
side 1-1.6	front-black 1.2	30	16384
side 2-0.5	front-black 1.5	30	16384
side 2-0.8	front-black 1.6	30	16384
side 2-1.2	front-yellow 0.5	30	16384
side 2-1.6	front-yellow 0.8	30	16384
	front-yellow 1	30	16384
	front-yellow 1.2	30	16384
	front-yellow 1.6	30	16384

B. *Data Wrangling*

To organize and analyze our data, we used pandas, as a software library written for the python programming language. The name is derived from the term “panel data”, an econometrics term for data set that include observations over multiple time periods for the same individuals. One of the library features is DataFrame object for data manipulation with integrated indexing. [1]

At first, we import data from each files of object 1 and object 2 are provided by function with prefix “read”. As an output it showed 16385 columns, or we want to achieve 16384. In this matter we reshaped it into 16384 then we drop the oversize data. After that, we transpose data into rows and concatenate 30 csv-file into one file as it is shown below. Finally, we convert all the data from string to float. (label)

Figure 1 : Merging files.



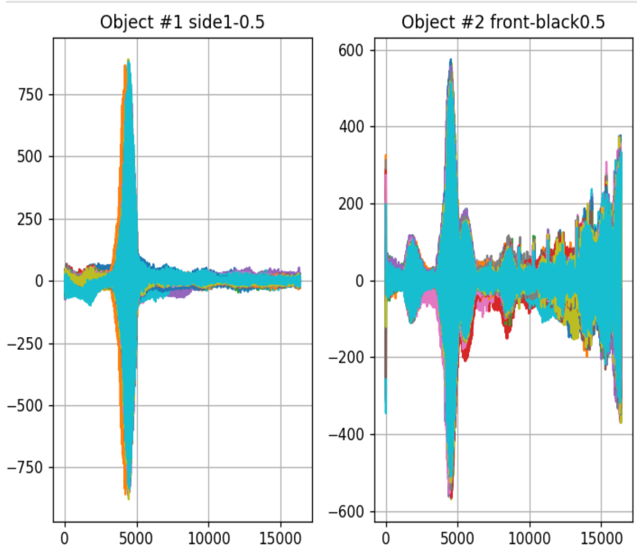
	001	002	003	004	005	006	007	008	009	010	...	021	022	023	024	025	026	027
0	-5.000	-12.000	4.000	-48.00	-8.000	-10.000	-2.000	1.000	-4.000	-4.000	...	9.000	0.000	-5.000	-5.000	9.000	6.000	10.000
1	0.000	-11.000	5.000	-47.00	-6.000	-8.000	-4.000	1.100	-3.000	-4.100	...	9.100	4.000	-2.000	0.000	6.000	6.100	10.100
2	1.000	-5.000	6.000	-53.00	-4.000	-4.000	-5.000	2.000	-7.000	-3.000	...	9.200	5.000	-4.000	6.000	3.000	-47.000	5.000
3	2.000	-2.000	6.100	-45.00	-3.000	-1.000	-8.000	7.000	-1.000	11.000	...	10.000	13.000	-4.100	8.000	2.000	-41.000	1.000
4	-15.000	-35.000	6.200	-40.00	0.000	-59.000	28.000	-62.000	49.000	20.000	...	18.000	-53.000	-6.000	-33.000	-23.000	-36.000	45.000
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
16379	6.400	-7.422	4.479	-48.35	-6.247	-6.240	4.413	-1.570	3.503	1.501	...	7.405	2.573	-9.357	5.430	-5.540	-3.493	-1.537
16380	3.547	-3.539	4.480	-49.30	-9.177	-4.271	1.520	-5.478	8.351	1.502	...	12.246	1.530	-8.349	3.487	-2.515	-1.550	-4.499
16381	-1.515	-7.423	4.481	-49.31	-8.198	-6.241	0.547	-2.572	9.299	1.503	...	14.186	1.531	-7.434	-1.584	1.572	0.570	-5.491
16382	-6.467	-7.424	-2.569	-50.43	-7.241	-6.242	5.431	2.528	5.489	-3.565	...	13.231	0.549	-7.435	-6.492	5.463	3.550	5.494
16383	-6.468	-10.326	-2.570	-50.44	-5.293	-2.291	6.400	-1.571	1.518	0.462	...	12.247	-1.539	-7.436	-7.486	6.451	8.399	10.287

16384 rows × 30 columns

C. *Distingish between two classes*

On this part, we want to distinguish between "object#1" and "object#2" after reshaping all data. We start by creating a new label for each object, then we plot it to see how the data looks like. In the figure bellow, we took a sample data from "side1-0.5" object#1, and "front-black 0.5" object#2, it showed us the reflecting object "1 & 2".

Figure 2 Plotting object#1 and object#2.



III. Creating a Binary Classifier

A. *Convolutional neural network (CNN)*

The convolutional neural networks (CNNs, [2] [3]) have become state-of-the-art models in the computer vision community [4]. In speech recognition, the CNNs have been applied in the same fashion to critical band energies (CRBE), obtained by a short-term filter bank [5] [6] [7]. We first consider a fully connected hidden layer. Given a sequence of D-dimensional observations arranged in a mini-batch of length N, the output of a hidden layer y at node i and time n is calculated from the input $x_n \in \mathbb{R}^D$ as

$$y_{i,n} = \sigma\left(\sum_{j=0}^{D-1} w_{ij} x_{j,n} + b_i\right), \qquad 0 \le n \le N \qquad (1)$$

where $w_{i,j}$ are the weights, b_i is the bias and σ is a non-linear function such as sigmoid or rectified linear unit [8]. The free parameters ($w_{(i,\bullet)}, b_i$) depend on the identity of the neuron i. In a typical convolutional layer that operates on both time and frequency dimensions (i.e. columns and rows of the mini-batch matrix), the weights are arranged as a 2D-patch and the parameters $w_{(i,\bullet,\bullet)}, b_i$ also depend on the identity of the convolutional unit i. The output of such a neuron can be evaluated at different positions (m,n):

$$y_{i,m,n} = \sigma\left(\sum_{j=m}^{m+k-1} \sum_{h=n}^{n+k'-1} w_{i,j-m,h-n}\, x_{j,h} + b_i\right) \quad (2)$$

where the weight patch $w \in \mathbb{R}^{k \times k'}$ $(k < D, k' < N)$ is multiplied with a part of the input covering the direct neighborhood of the position (m,n). The set of outputs of a convolutional unit i at all positions $\{y_{i,\bullet,\bullet}\}$ is referred to as "feature map" [2] and can be thought of as a feature stream, extracted by this unit. For brevity, we omit the discussion on patch symm [4]etry and boundary handling. However, within the framework of framewise training it is convenient to end up with a feature map that has the same number of columns as the mini batch.

Convolutional layers need to support processing of multiple feature streams for two reasons. First, the input can consist of pre-processed features like log-Mel energies and their first/second temporal derivatives. Second, once we stack more than one convolutional layer, the output of each filter from the first layer *l-1* is considered an input stream for the following layer *l*. If the R input streams are also arranged in mini batches of the same dimension, the weight patch and the input both get an additional dimension that needs to be summed over in Eq. 2: [9]

$$y_{i,m,n} = \sigma\left(\sum_{j=m}^{m+k-1} \sum_{h=n}^{n+k'-1} \sum_{r=0}^{R-1} w_{i,j-m,h-n,r}\, x_{j,h,r} + b_i\right) \quad (3)$$

Convolving the raw speech signal in time is a one-dimensional operation on a single input stream, which simplifies Eq. 3 as follows. First, the mini-batch is now constructed from consecutive signal intervals of length D (*e.g.* $D = 170ms \,.\, 16kHz = 2720$). The output of a convolutional unit i is then defined as

$$y_{i,m} = \sigma\left(\sum_{j=m}^{m+k-1} w_{ij-m} x_j + b_i\right) \quad (4)$$

where m is the position within the output vector and $\omega \in R^k$ is a weight vector. Evaluating the inner product in

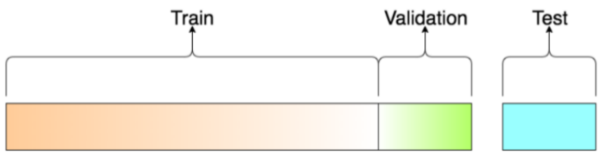
Eq. 4 for each sample *m = 0,1,2,..* would be very expensive and the output would be highly correlated. For this reason, it is useful to proceed with a step size *s > 1*, such that the output feature stream is only calculated for a subset of positions: $\{y_{i,m.s} : 0 \leq m.s < D - k\}$. [9]

If *s = 1*, the hidden unit i performs the convolution of the input vector with the mirrored weight vector (denoted as $\widetilde{w}_i$) $\sigma(x * \widetilde{w}_i + b_i)$, producing an output sequence of length *D-k+1*. If s > 1, the output of the hidden unit *i* is sub-sampled by this factor after evaluating convolution. Stacking multiple 1Dconvolutional layers is possible by introducing the additional dimension to the weight and the input vector analogously to Eq. 3, such that $x^{(l)}_{\bullet,r} = y^{(l-1)}_{r,\bullet}$ for two consecutive layers *l-1* and *l*. A convolutional layer that consists of G filters has o [2] [3]nly $G \cdot (Rk + 1)$ trainable parameters, which is usually far less than in a fully connected layer.

B. Spliting Data

We upload our data from object #1 and object #2, then we start preparing process. In this step we split the cleaned Data into 3 parts as it shown on the Fig 3: First, Training Dataset: the sample of data used to fit the model. This dataset that we use to train the CNN model (train_data consist 60% of our data). The model will go through and learns from this data. Then, we create an equal portion Y_train (342,) time signal representing the labels (0, 1). Second part, Validation Data: this portion of data is used to provide an unbiased evaluation of a model fit on the training dataset while tuning model hyperparameters. The evaluation becomes more biased as skill on the validation dataset is incorporated into the model configuration. The validation set used to evaluate our model (validation_data we took 20% of our data) plus we create y_validation “labels” of equal amount. Hence the model never sees this data in the training, but it does learn from this data. As a validation set result (114,)ts. The third part, Test Dataset: the sample of data provide an unbiased evaluation of a final model prediction already trained on the previous datasets. The test dataset performs the gold standard to evaluate the model. To recap, we needed to train our data, we used a Cross validation (test_data with 20%) as a result (342,) time signal. In the next step, we reshaped our data parameters into a canonic form: x_test (114,1,16384,1),x_validation (114,1,16384,1) and x_train (342,1,16384,1) and reshaping Y_train data to (114,2) for convenient input model.

Figure 3 : Visualization of the splits



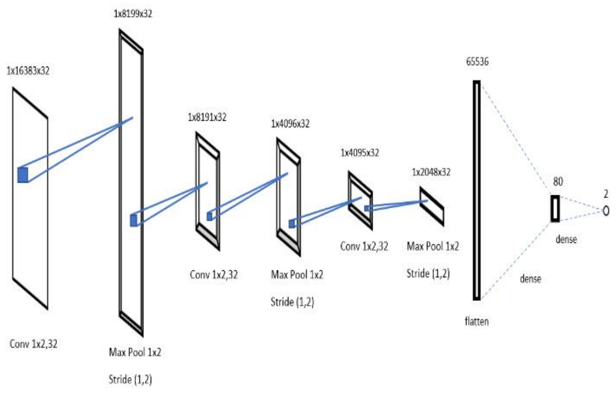
C. Creating a descriminator

To discriminate the reflected sound signal, CNN is designed with one input layer, 3 convolutional layer and one output layer as demonstrated in fig 4.

The input layer consists of the matrix (1, 16384, 1). The first layer is the convolutional layer 1 (conv1) mainly responsible for feature extraction from the input data. This layer also accomplishes convolution operation to the tiny, contained areas by convolving a feature map with the preceding layer. In addition, it consists of multiple feature maps with learnable kernels and rectified linear units (ReLU). The kernel Size determines the locality of the filters. At the end of each convolution layer and fully connected layer, ReLU is utilized as an activation function to ameliorate the performance of the model. Next layer is the Max pooling is used to subsample each feature map. For converting the 2D featured map matric to a 1D feature vector, a flatten layer is used after the Max pooling. Another hidden layer is the fully connected layer is also known as the dense layer. It is similar to the hidden layer of Artificial Neural Network (ANNs) but here it is fully connected which attaches every neurons of the previous layer to the next layer. [10] In order to reduce overfitting, dropout regularization method is used at fully connected layer 1 which randomly switches off some samples during the training to argument the performance of the system. The final layer is a softmax layer with 2 neurons as we have 2 objects of output sounds.

We compile the model using the Adam optimizer and measuring loss using cross entropy loss due to a classification problem.

Figure 4 CNN Model



D. Asses the classifier

Before we start assessing the classifier we Generate A class prediction for the input samples batch by batch. We use as Arguments x as input data as list of Numpy arrays and a batch size. That will return a number defining the predicted class.

Classification is one of the supervised learning techniques, in which the labels are discrete groups that classify individual data points. Based on what it has known about historical records, the classificator is most likely for new data. The estimated value can be tested against the real (i.e. the basic truth) mark to measure the precise nature of the model since the data is labelled entirely. There are only two labels ("normal" and "abnormal") for a binary classification device

a) True Positive (TP)

Sensitivity (True Positive rate) measures the proportion of positives that are correctly identified (i.e. the proportion of those who have some condition (affected) who are correctly identified as having the condition). [11]
As result in our experiment TP: [47 66]

b) True Negative (TN)

Specificity (True Negative rate) measures the proportion of negatives that are correctly identified (i.e. the proportion of those who do not have the condition (unaffected) who are correctly identified as not having the condition). [11] As result in our experiment TN: [66 47]

c) False Positive (FP)

A False Positive is the incorrect identification of anomalous data as such, i.e. classifying as “abnormal” data which is in fact normal. [11] As result in our experiment
FP: [0 1]

d) False Negative (FN)

A False Negative is the incorrect identification of data as not being anomalous, i.e. classifying as “normal” data which is in fact abnormal. [11] As result in our experiment FN: [10]

e) False Discovery rate (FDR)

The False Discovery Rate (FDR) is the proportion of hypotheses that we falsely think are true. In our case, the FDR is the faction of the feature weights that we think are not zero, even though they really are, in fact, zero.
For a single hypothesis test, we can compute a p-value: the probability that we say the hypothesis is true even though it is false. [11] As result in our experiment FDR: [0 0.01492537]

$$FDR = \frac{FP}{(TP + FP)} \qquad (5)$$

f) Negative Predictive value (NPV)

is defined as the proportion of predicted negatives which are real negatives. It reflects the probability that a predicted negative is a true negative. [11] As result in our experiment NPV: [0.98507463 1]

$$NPV = \frac{TN}{(TN + FN)} \qquad (6)$$

g) True Positive Rate (TPR)

“Sensitivity”, is a measure of the proportion of positive cases in the data that are correctly identified as such. It is defined in eq. 1 as the total number of correctly identified positive cases divided by the total number of positive cases (i.e., abnormal data): [11]
As result in our experiment TPR: [0.97916667 1]

$$TPR = \frac{TP}{(TP + FN)} \qquad (7)$$

h) Fall out or false positive rate (FPR)

Is the proportion of negative cases incorrectly identified as positive cases in the data (i.e. the probability that false alerts will be raised). It is defined in eq. 2 as the total number of negative cases incorrectly identified as positive cases divided by the total number of negative cases (i.e. normal data). [11] As result in our experiment FDR: [0 0.020833333]

$$FPR = \frac{FP}{(FP + TN)} \qquad (8)$$

i) True Negative Rate (TNR)

Specificity”, is simply 1 minus the **FPR**. [11] As result in our experiment TNR : [1 0.97916667]

$$TNR = \frac{TN}{(FP + TN)} \qquad (9)$$

j) F1 score

Is defined by:

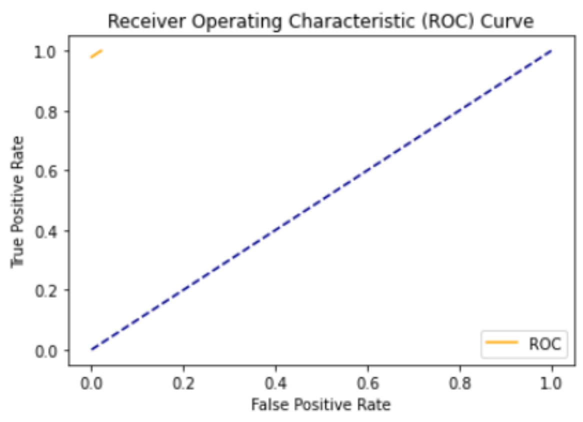
$$F1 = \frac{TP + TN}{TP + TN + FP + FN} \qquad (10)$$

As result in our experiment F1 score: 0.99124812030075187

k) ROC curve

AUC–ROC curve is the model selection metric for bi–multi class classification problem. ROC is a probability curve for different classes. ROC tells us how good the model is for distinguishing the given classes, in terms of the predicted probability. A typical ROC curve has False Positive Rate (FPR) on the X-axis and True Positive Rate (TPR) on the Y-axis. [12]

Figure 5: ROC curve



The area covered by the curve is the area between the orange line (ROC) and the axis. This area covered is AUC. The bigger the area covered, the better the machine learning models is at distinguishing the given classes. Ideal value for AUC is 1.

IV. Grafical user interface (GUI)

After we solve the problem, we want other people to use the model that we built without the help of python library, for that reason we deploy our result in a graphical user interface as web application by utilizing anvil. [13]

A. *Anvil*

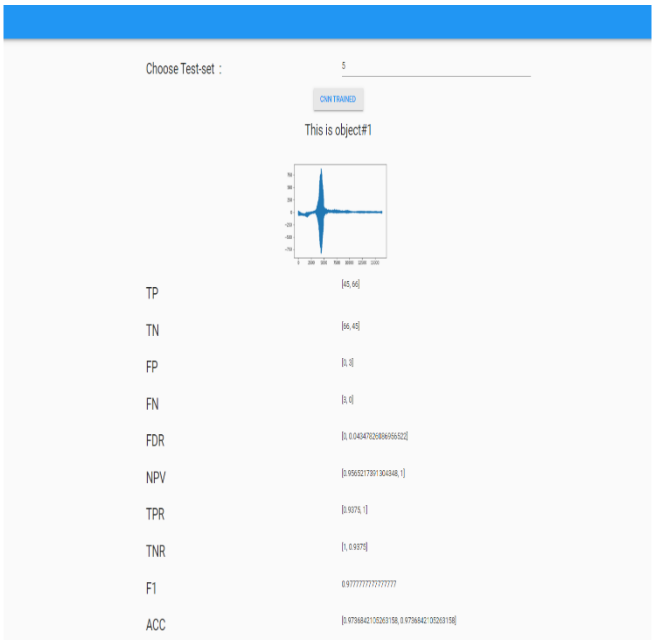
Is a new way to build web apps, with nothing but Python. An anvil application is made up of:

- A User Interface, which we design with a drag-and-drop designer.
- Client-side Python code, which runs in the web browser.
- Server-side Python code, which runs on Anvil's servers.
- A built-in database (Data Tables), which stores our data.
- Some Python code running on computer, which can also interact with our application.

B. *Simulation*

Once we build our user interface, we connect it to our notebook by enabling the uplink in the anvil editor and paste the uplink-key to our program in notebook. The we start simulation our application by choosing the Test-set. If we chose any number between [0 48] the solution would be object #1 as we can sees in fig6 but if we give a Test-set number in the range [48 113] it will be object#2 fig 7

Figure 6: Simulating Object #1

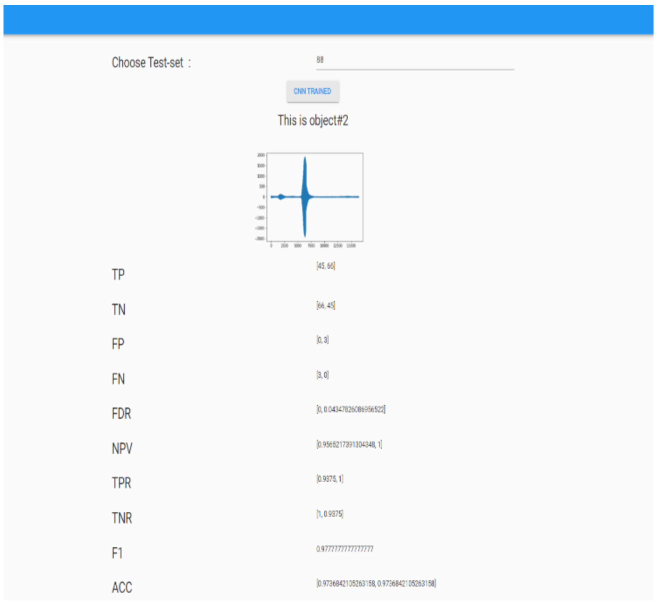


Choose Test-set : 5

This is object#1

TP	[45, 66]
TN	[66, 45]
FP	[0, 3]
FN	[3, 0]
FDR	[0, 0.043478260869565522]
NPV	[0.9565217391304348, 1]
TPR	[0.9375, 1]
TNR	[1, 0.9375]
F1	0.9777777777777777
ACC	[0.9736842105263158, 0.9736842105263158]

Figure 7: Simulating Object #2



Choose Test-set : 88

This is object#2

TP	[45, 66]
TN	[66, 45]
FP	[0, 3]
FN	[3, 0]
FDR	[0, 0.043478260869565522]
NPV	[0.9565217391304348, 1]
TPR	[0.9375, 1]
TNR	[1, 0.9375]
F1	0.9777777777777777
ACC	[0.9736842105263158, 0.9736842105263158]

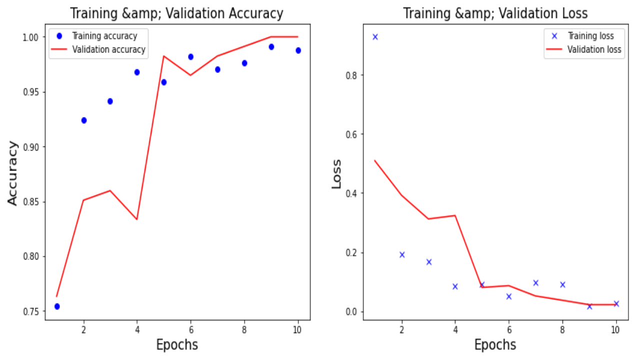
V. Result

After training the model with "X_train" and "X_Validation" on 10 epochs, and with a "batch_size" equal to 5, the model gives a final accuracy of 0.9912280, this result is based on a strange data set "X_test" to the training set, and the loss value reached:0.0295263, in the training process the model took on each epoch an average of 18 second/step, starting with a Training loss: 1.3782, and Validation loss: 0.4257, then the Training loss dropped to 0.2637 and continued fluctuation between 0.1135 and 0.05 until stabilize on 0.0042. Meanwhile, the Validation loss was bigger in the first 6 epochs then dropped to 0.0496, as for Training accuracy start mediocre Training accuracy: 0.6989 then elevating to 0.9303 and ending 0.9988, in the other hand

Validation accuracy took a slower development 0.7982 and ending by 0.9649 as shown in the plot below.

To epitomize the final accuracy and Training accuracy, Validation accuracy is contiguous so we can deduce that our model can sustain a real-world data, in other words, is scalable.

Figure 8 : the model accuracy vs Epochs



VI. References

[1]"WIKIPEDIA,"[Online].Available:https://en.wikipedia.org/wiki/Pandas_(software)#:~:text=pandas%20is%20a%20software%20library,for%20data%20manipulation%20and%20analysis.&text=The%20name%20is%20derived%20from,periods%20for%20the%20same

[2] B. B. J. S. D. D. H. R. E. H. H. a. L. D. J. Y. LeCun, "“Handwritten digit recognition with a back-propagation network,”," vol. vol. 2., no. in Advances in Neural Information Processing Systems 2, D. Touretzky, Ed.,, Denver, CO: Morgan Kaufman, 1990.

[3]T. H. G. H. K. S. a. K. L. A. Waibel, “Phoneme recognition using time-delay neural networks,”, Acoustics,Speech and Signal Processing,IEEE Transactions on,vol. 37, no. 3, pp. 328–339,, Mar. 1989.

[4] I. S. a. G. E. H. A. Krizhevsky, "“ImageNet classification with deep convolutional neural networks,”," *Curran Associates,* Vols. F. Pereira, C. Burges,L. Bottou, and K. Weinberger, Eds., no. in Advances in Neural Information Processing Systems 25,, p. pp. 1097–1105., 2012,.

[5] A. M. H. J. a. G. P. O. Abdel-Hamid, "“Applying convolutional neural networks concepts to hybrid NN-HMM model for speech recognition,”," *in Proc. IEEE Int. Conf. on Acoustics,Speech and Signal Processing,Kyoto, Japan,,* p. pp.4277–4280., Mar.2012.

[6]H. K. L. M. G. S. a. T. B. H. Soltau, "“Neural network acoustic models for the DARPA RATS program,”," *in Proc Interspeech, Lyon, France,,* p. pp. 3092–3096., Aug. 2013.

[7] B. K. A. M. G. E. D. G. S. H. S. T. B. A. Y. A. a. B. R. T. N. Sainath, "“Improvements to deep convolutional neural networks for LVCSR,”," *in Proc. IEEE Automatic Speech Recognition and Understanding Workshop (ASRU), Olomouc, Czech Republic,,* p. pp.315–320., Dec. 2013

[8] V. Nair and G. E. Hinton, "“Rectified linear units improve restricted Boltzmann machines,”," *in Proc. of the 27th Int. Conf. on Machine Learning, Haifa, Israel,,* p. pp. 807–814., Jun. 2010,.

[9] Z. T. R. S. H. N. Pavel Golik, "Convolutional Neural Networks for Acoustic Modeling," *INTERSPEECH,* p. 1, 2015

[10] S. S. M. B. S. Fathma Siddique, "Recognition of Handwritten digit using convolutional neural network in python with tensorflow and Comparison of performance for various hidden layers," p. 542, 26-28 September.

[11] "PICO Knowkedgebase," 2021. [Online]. Available: https://www.pico.net/kb/what-is-a-false-positive-rate.

[12] "WIKIPEDIA,Receiver operating characteristic," [Online].Available:https://en.wikipedia.org/wiki/Receiver_operating_characteristic#:~:text=A%20receiver%20operating%20characteristic%20curve,why%20it%20is%20so%20named..

[13] "Anvil Docs," [Online]. Available: https://anvil.works/docs/overview.