International Gang - Convolution Neural Networks

Ahmed Shahhat and Nguyen Xuan Tung
(Dated: May 12, 2021)

In this paper, we discuss about the application of Convolution Neural Networks in training the data with two different type of data set. We start training the network with the generated data with reduced signal-to-noise ratio to find the failure of discriminating the categories. We then discuss the best performance of Convolution Neural Networks together with the regularization through a given data set. The results show that the amplitude of the external signal in the data consequences the network and Lasso Regression makes the network perform best.

INTRODUCTION

In our modern world, machine learning is one of the most exciting and dynamic areas of modern research and application which focuses on the study of computer programs that can access data. One of the most significance field in machine learning is Neural Networks, especially Convolution Neural Networks (CNNs).

Before going deeper into our subject, we provide a basic introduction about the CNNs and its principles. CNNs is the special form of neuron network in which the feature of filter/mask is to learn to detect abstract concepts. In principle, a neural network consists of many such neurons stacked into layers, with the output of one layer serving as the input for the next [1]. The first layer is usually called the input layer, the middle layers are called "hidden layers", and the last layer is called the output layer. By stacking layers of convolutions on each other, we can make more conceptual and more information from a CNN. In general, the mathematics behind this is the matrix multiplication. There are two kinds of basic layers that make up a CNN: convolution layers and pooling layers. These layers are used to reduce the dimension of a layer output through operations. FIG 1 shows us the overall idea of this structure.

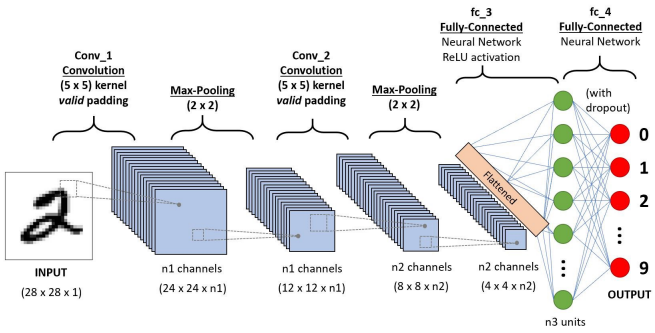

FIG. 1: The structure of Convolution Neural Network [2]

In order to prevent the overfitting in CNNs, regularization is introduced. Essentially, it is a method that controls the model complexity. In another word, they are used to avoid the filters of the network to become non-zero. For instance, in our work we expect these bumps is recognised by the filter, we could expect 1 or couple of filters are enough to recognise them. By introduce the regularization, which cut off many weights when they are not necessary. The cost function of regularization can be expressed as:

$$C_\theta \to C_\theta + \lambda R_\theta$$

Where λ represents for the multipliers, tunes strength of regularization and θ_i is one of the weights in hyperparameters. There are two main regularization, L1 and L2. L1 regularization or also known as Lasso Regression, which a penalty equal to the absolute value of the magnitude of coefficients. The function of L1 is showed as: $R_\theta = \sum_i |\theta_i|$.

L2 regularization is called Ridge Regression, it is a model tuning method that is used to analyse data when it has the correlations between predictor variables.The function of L2 is showed as: $R_\theta = \sum_i |\theta_i|^2$.

The key different between these two technique is that Lasso shrinks the less important feature's coefficient to zero thus, removing some feature altogether. So, this works well for feature selection in case we have a huge number of features [3].

METHODS

The very first task in our work is to generate the data with some notation:

$$x_j = \{x(t_1), \ldots, x(t_L)\} \quad (1)$$

$$\hat{x}_j = \{\hat{x}(t_{L-i+1}), \ldots, \hat{x}(t_{L-i+Z})\} \quad (2)$$

The j^{th} is the input time-series without the bump and the bump itself. The parameter Z and L are corresponding to the bump lengths and the time series, respectively, where there values are 12 and 60 (i is the range from Z to L).

$x_j(t)$ are the signals which is generated by sampling the amplitude different dx between $x_j(t_i)$ and $x_j(t_{i+1})$. The distribution of this signal is:

$$P(\mathrm{d}x) \sim \exp\left(-\frac{|\mathrm{d}x - b|}{\Delta x}\right) \tag{3}$$

where b is a parameter called bias and Δx is the step typical size of the jump process. In this task, we set them to be 5 and 50, respectively. From equation (3), we can get the sampling rule by applying the inverse transform:

$$\mathrm{d}x = [\log(p)\Delta x] \cdot 2\,\mathrm{sign}(q - 0.5) + b \tag{4}$$

where random number sampled from a uniform distribution p is in [0,1], value q is chosen in 0,1 with the probability is equal for all element assigned of the set.

$\hat{x}_j(t)$ signals are sine bumps generated through:

$$\hat{x}_j(t_{i+k}) = [A \cdot \sin\left(\frac{\pi \cdot k}{Z}\right)] \tag{5}$$

where A is the amplitude of the signal, which is fixed to 500. In this task, we reduce the amplitude to 200 to see the network starts to fail discriminating the categories.

Finally, we have the the possibilities for the input signals, which is labeled with variable y:

$$x = \begin{cases} x_j + 0 \cdot \hat{x}_j & y_j = 0 \\ x_j + 1 \cdot \hat{x}_j & y_j = 1 \\ x_j - 1 \cdot \hat{x}_j & y_j = 2 \end{cases} \tag{6}$$

In the second task, we start the CNNs implementation for the given data set of the second task. The network is built with the help of Python libraries: Keras and TensorFlow, which allow us to implement the work of the CNNs in a few lines of code. We divide this part of the work into 2 other small steps:

- in the first step, we change the parameters and layers in CNNs to check the performance of the model and to find the factor affect the network. In order to do that, we improve the CNNs code where only one convolution layer, and five dense layers are included. Each dense layer is included with different activation function: relu, sigmoid, tanh, and softmax respectively. Moreover, the trainable parameters are set to be 596.

- in the second step, we find the best regularization applied to the model. We start to test the performance of CNNs with the Lasso regularization (L1), Ridge regularization (L2) and the combination of Lasso/Ridge regularization (L1/L2). The weights in the filters are also visualized for each CNNs performance.

RESULTS

In the first task, we perform a confusion matrix, which is essentially for the prediction. The absent in the confusion matrix represents value = 0, and so on for the positive and negative represented for positive value and negative value, respectively. By visualize the color on the confusion matrix, we can see the level of prediction of the network. As we can see in the FIG 2 (top figure), the most errors are located in "absent-absent", "negative-negative", and "positive-positive" cells, which shows that everything is working correctly.

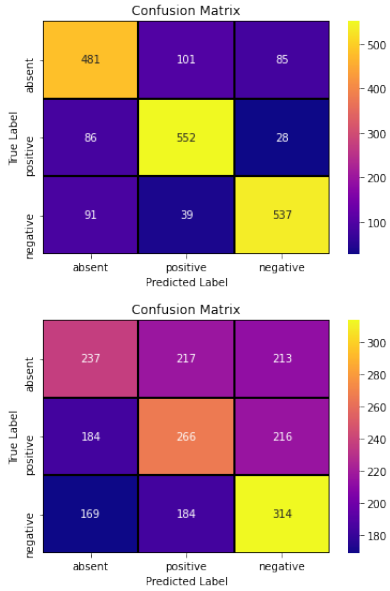

FIG. 2: Confusion matrices. On top, the confusion matrix of original data. On bottom, the confusion matrix of reduced data

After showing the network is working properly, we start to reduce the signal to noise ratio of the network. We test multiple values start from 100 to 500, we can see that by reducing the amplitude, the model is getting worse and worse. As the results, the model is able to make a good predictions from a signal of amplitude 350 and larger. The bottom figure of confusion matrix in FIG 2 shows that, the most errors are not located in the correct cells anymore, which means the model starts to fail discriminating the categories. When the value of signal to noise goes down to 200, we can observe from the FIG 3 that, in the second figure of the accuracy and loss function, the accuracy of the training data and validation data is around 0.4, which is equivalent to a random guess. From this we can conclude that by reducing the signal to noise to 100, the model is completely random.

In the second task, the given data is compiled, which mean we can not know what inside the data. In this data, there are 3 labels: 0, 1, and 2. After training the model

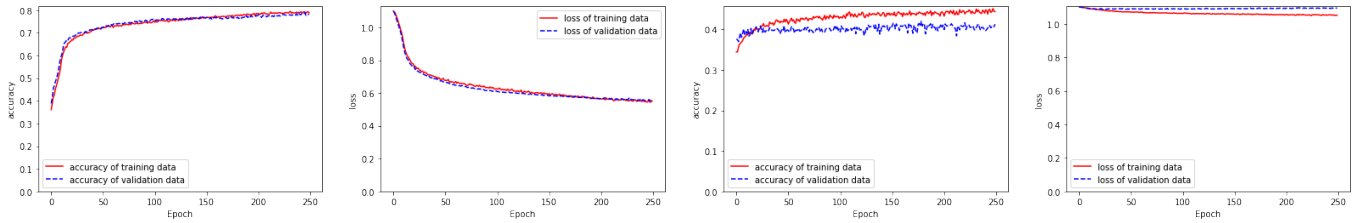

FIG. 3: Accuracy and loss function of the original data (on left) compared with the reduced data(on right)

with a improved network, we can see that by adding more Dense layers with maximum 600 trainable parameters, the model runs slightly quicker than the old model. The FIG 4 shows us the performance and the detail architecture of the trained model. By adding more Dense layers and removing 1 Convolution layer, we can observe that the factor that affect the result tend to be the number of layers rather than number of parameters.

Layer type	Output shape	Param
Conv1D	(None, 50,5)	60
Av. Pool.	(None, 10, 5)	0
Flatten	(None, 50)	0
Dense	(None, 8)	408
Dropout	(None, 8)	0
Dense	(None, 8)	0
Dense	(None, 6)	54
Dense	(None, 5)	35
Dense	(None, 5)	24
Dense	(None, 3)	15

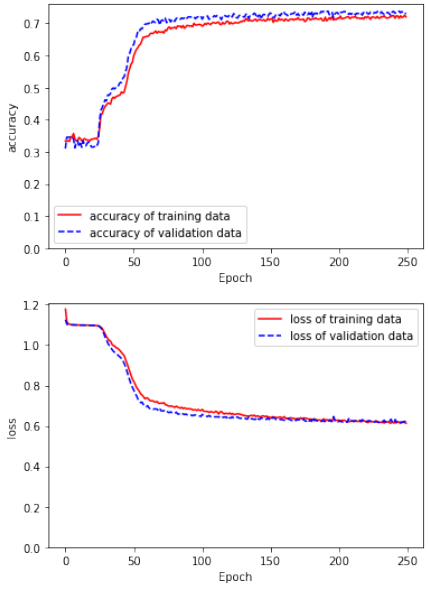

FIG. 4: Architecture of the improve model and results of accuracy and loss of the model.

In the FIG 5, we can see clearly the change of the weights filter according to the different regularization applied. Comparing the weights applied with L1 and L2, we can see that there is a slightly increasement of the weights in L2, but the first filter is not working properly in L2 regularization. On the other hand, we observe that a significant reduction of the weights in the combination of L1 and L2. This means that the results are converted into a stable predictor for each regularization. We can conclude that the best regularization shown to be L1 (Lasso) regularization.

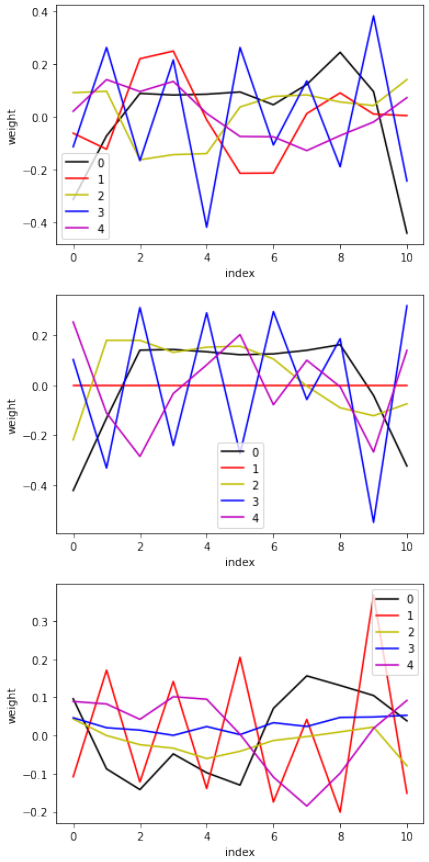

FIG. 5: Weight of the model. The model applied with L1 regularization, L2 regularization and L1 L2 regularization from up to down respectively.

CONCLUSION

In this work, we show that the implementation of the Convolution Neural Network is able to work correctly

with the generated data and the given data. The best way is to study the structure of the data is to decrease the signal to noise ratio. It shows us that the model strongly depends on the amplitude of the external signal in the data. We see that the model is stable and able to make good predictions in high amplitude value A. Another aspect of our work is the regularization. The study shows that regularization makes the improvement to the network. For different regularization, we have different results and its affect the filter weights of the model. By applying the different regularization to the model, we find that L1 (Lasso) regularization to be the best choice among others.

[1] Pankaj Mehta , Marin Bukov , Ching-Hao Wang, Alexandre G.R. Day, Clint Richardson , Charles K. Fisher, David J. Schwab. (2019). et al., *"A high-bias, low-variance introduction to Machine Learning for physicists"*. In Phyics Reports 810 (2019) 1-124.

[2] Sumit Sasha 2018, A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way, towards data science, viewed 4 May 2021, https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

[3] Anuja Nagpal 2017, L1 and L2 Regularization Methods, towards data science, viewed 5 May 2021, https://towardsdatascience.com/l1-and-l2-regularization-methods-ce25e7fc831