Using logic rules to achieve interpretable Convolutional Neural Network

Vahid Bahrami Foroutan[a] and Elham Bahrami Foroutan [b]

[a] Higher Educational Complex of Saravan, Iran.

[b] University of Bremen, Germany.

Article History: Received: 11 January 2021; Revised: 12 February 2021; Accepted: 27 March 2021; Published online: 10 May 2021

Abstract: Using logic rules in the Convolutional Neural Network (CNN) is helpful of CNN. The motivation of our paper is to show that there is a possibility to turn black box to white box. Moreover, in the proposed methodology, the output and output production (convolutional formula) will be interpretable. In our paper, it is shown that it would be possible to go from LCNN to CNN and vice versa. For this reason, score function is developed using quantum logic formula. Therefore, it is proven that there are some rules between input and output and the way of output production could be interpreted by the rules. This rules help us to understand CNN method.

Keywords: Logic, CNN, LCNN, Score function

1. Introduction

CNN provides a useful mechanism for learning patterns. However, the problem of CNN is that its results are not interpretable for human. Thus, using logic rules would be beneficial in order to achieve interpretable CNN. Some relevant papers are investigated to use rules in CNN. In [1], a CNN-based deep learning structure is applied as the fundamental building block of a data-driven classification network. A FL (Fuzzy Logic) based layer is presented as an integral part of the overall CNN structure which acts as the main classification layer of the deep learning structure. One could extract directly from the deep learning structure linguistic rules in the form of a FL rule-base. Reference [2], presents a system called Churn-teacher using an iterative distillation method which transfers the knowledge, elicited using just the combination of three logic rules ($X\&Y = max\{X + Y\text{-}\ 1.0\}$, $X \vee Y = min\{X + Y. 1\}$, $X_1 \wedge \ldots \wedge X_N = \sum_i \frac{X_i}{N}$) into the weight of Deep Neural Networks. Authors investigate the use of CNNs augmented with structured logic rules to overcome or reduce uninterpretability. Authors used the mean for assessing results. In [3], a novel critic learning based deep convolutional neural network framework is presented to address drawbacks. Papers [4-6] also focused on CNN drawbacks papers. All above mentioned papers suffer from two deficiencies. First, the prior-knowledge given may be incorrect sometimes. Although, the methods do not have influence on the knowledge rules adaptively. Second, most of the existing studies regard only simple knowledge rules, and more sophisticated structures are ignored or not carefully used. In [3] their work is a two-branch CNN model, where one branch for creating a predictor from text features and the other one to train a predictor with the given knowledge rules. In our paper, score function is developed as an innovative work using quantum logic formula [10] where inputs scaled to interval $\in[0,1]$. In QL [10] an scheme is presented which applies conjunction between inputs based on weighting that we use it. For classification of images (Chair, Cat, Cow, Clothes) assume some features for input, output is the interpretation of results (class of assumed input). For example, there are some pictures and the purpose is to find a cat, inputs are some features and output is the interpretation of cat. According to the Fig. 1, there are four classes.

Figure 1. Classification of images(chair, cat, cow, clothes).

2. Problem

The problem is that CNN is a black box. The motivation of our paper is to show that there is a possibility to turn black box to white box. Moreover, in the proposed methodology, the output and output production (convolutional formula) will be interpretable. In our paper, it is shown that it would be possible to go from LCNN to CNN and vice versa.

3. Proposed solution

For finding a solution, it must be proven that there are some rules between input and output and the output production could be interpreted by the rules and proven that there is a relation between input and output. This rules help us to understand CNN method.

4. Idea of solution

We use these rules in our work :

Definition 1: Let $\mu_1.\mu_2$ be CQQL conditions and $\Theta = \{\Theta_1.\ldots.\Theta_n\}$ with $\Theta_i \in [0,1]$ a set of weights [10].

$$\wedge_{\Theta_1.\Theta_2}(\mu_1.\mu_2) \Rightarrow (\mu_1 \vee \neg\Theta_1) \wedge (\mu_2 \vee \neg\Theta_2) \quad (1)$$

$$\vee_{\Theta_1.\Theta_2}(\mu_1.\mu_2) \Rightarrow (\mu_1 \wedge \Theta_1) \vee (\mu_2 \wedge \Theta_2) \quad (2)$$

Definition 2: Let $\mu(o_1.o_2)$ be a CQQL condition on objects $o_1$and o_2 in the required syntactical form, then $eval1(\mu(o_1,o_2))$ is recursively defined as [10, 11]:

$$eval1(\mu_1(o_1.o_2) \wedge \mu_2(o_1.o_2)) = eval1(\mu_1(o_1.o_2)).eval1(\mu_2(o_1.o_2)) \quad (3)$$

$$eval1(\mu_1(o_1.o_2) \vee \mu_2(o_1.o_2)) = eval1(\mu_1(o_1.o_2)) + eval1(\mu_2(o_1.o_2)) - eval1(\mu_1(o_1.o_2)).eval1(\mu_2(o_1.o_2)) \quad (4)$$

$$eval1(\mu_1(o_1.o_2) \vee \mu_2(o_1.o_2)) = eval1(\mu_1(o_1.o_2)) + eval1(\mu_2(o_1.o_2)) \quad (5)$$

$$eval1(\mu_1(o_1.o_2)) = 1 - eval1(\mu_1(o_1.o_2)) \quad (6)$$

Equation 3, 4 are used when μ_1 and μ_2 are not exclusive and equation 5 is used when μ_1 and μ_2 are exclusive.

Definition 3: According to [10] we use this formula:

$$d \wedge_{\Theta_1.\Theta_2} f = (d + (1-\Theta_1) - d(1-\Theta_1))(f + (1-\Theta_2) - f(1-\Theta_2))$$

where f, d are conditions and Θ_1, Θ_2 are weights. First, the idea is to develop score function (in CNN, $f(x) = w * X + b$ is called score function where X is an input matrix and w is a weight matrix and b is a bias matrix) as an innovative work using quantum logic formula [10] (Definition 1 and 2 and 3) where inputs scaled to interval$\in$ [0,1]. At the beginning assume two inputs x1, y1 $\in$ [0,1], where x1, y1 are features of images and a weight matrix $w = \{w_{ij}\} \in [0,1]$ and i$\in$ {1.2}, j$\in$ {1.2}. Then we will create matrix X=$\begin{pmatrix} x1 & y1 \\ \neg x1 & \neg y1 \end{pmatrix}$. Following rules are used in this idea instead of $w * X$ in score function (according to Definition 1 and 2 and 3):

$$(x1 \wedge_{w_{11}.w_{12}} y1) = (x1 + (1-w_{11}) - x1(1-w_{11})) * (y1 + (1-w_{12}) - y1(1-w_{12})) \quad (7)$$

$$(\neg x1 \wedge_{w_{21}.w_{12}} y1) = ((1-x1) + (1-w_{21}) - (1-x1)(1-w_{21})) * (y1 + (1-w_{12}) - y1(1-w_{12})) \quad (8)$$

$$(\neg x1 \wedge_{w_{21}.w_{22}} \neg y1) = ((1-x1) + (1-w_{21}) - (1-x1)(1-w_{21})) * ((1-y1) + (1-w_{22}) - (1-y1)(1-w_{22})) \quad (9)$$

$$(x1 \wedge_{w_{11}.w_{22}} \neg y1) = (x1 + (1 - w_{11}) - x1(1 - w_{11})) * ((1 - y1) + (1 - w_{22}) - (1 - y1)(1 - w_{22})) \quad (10)$$

There are four conjunctions in our work (if we assume the bias is zero):

$$(x1 \wedge_{w_{11}.w_{12}} y1) = A \quad (11)$$

$$(\neg x1 \wedge_{w_{21}.w_{12}} y1) = B \quad (12)$$

$$(\neg x1 \wedge_{w_{11}.w_{22}} \neg y1) = C \quad (13)$$

$$(x1 \wedge_{w_{11}.w_{22}} \neg y1) = D \quad (14)$$

Then disjunction is applied between these minterms according to formula below (there is the arithmetic forms below for it, Because the purpose is to have a formula from disjunctive normal form):

$$A \vee_{w'_{11}.w'_{12}} B = (A * w'_{11}) + (B * w'_{12}) \quad (15)$$

$$\vee_{w'_{11}.w'_{12}.w'_{21}.w'_{22}} (A.B.C.D) = A * w'_{11} + B * w'_{12} + C * w'_{21} + D * w'_{22} \quad (16)$$

Where $w' = \{w'_{ij}\} \in [0,1]$ is the result of training of weighting (see section 6.1). Second, we will create a trained CNN and compare its output with the output of following logical formula to prove that there is a logical rule in CNN (see more details about trained CNN in section 7, table 3).

$$(a1 \wedge b1) = a1 * b1 \quad (17)$$

$$(\neg a1 \wedge b1) = (1.0 - a1) * b1 \quad (18)$$

$$(a1 \wedge \neg b1) = a1 * (1.0 - b1) \quad (19)$$

$$(\neg a1 \wedge \neg b1) = (1.0 - a1) * (1.0 - b1) \quad (20)$$

$$(a1 \vee b1) = a1 + b1 - a1 * b1 \quad (21)$$

$$\neg a1 = (1.0 - a1) \quad (22)$$

$$\neg b1 = (1.0 - b1) \quad (23)$$

Where a1, b1 are inputs in trained CNN (a1, b1 ∈ [0,1]). In this method, it is proven that it would be possible to get to LCNN from CNN and vice versa.

$$CNN \Leftarrow LCNN$$

5. Input and output

Input: Inputs are the features of the images.

Output: Output is the result of disjunction between below minterms (means output is the name of class according to input features):

$$(x1 \wedge_{w_{11},w_{12}} y1) = A \quad (24)$$

,

$$((\neg x1) \wedge_{w_{21},w_{12}} y1) = B \quad (25)$$

,

$$((\neg x1) \wedge_{w_{21},w_{22}} (\neg y1)) = C \quad (26)$$

,

$$(x1 \wedge_{w_{11},w_{22}} (\neg y1)) = D \quad (27)$$

6. Details of solution

The training data is according to table below. In this table there are four pictures with two features (color and texture). Color feature has two values, white and black and texture feature has two values, iron and non iron. There are four classes in this table: Chair, Cat, Cow, Clothes.

Table 1: Training data.

picture	color		texture		class
	white	black	iron	non iron	
1	0.19	0	0	0.8	Cat
2	0	0.22	0.72	0	Chair
3	0	0.35	0	0.67	Cow
4	0.20	0	0	0.70	Clothes

And the testing data is according to table 2. In this table there are four pictures with two features (color and texture). Color feature has two values, white and black and texture feature has two values, iron and non iron. There are four classes in this table: Chair, Cat, Cow, Clothes.

Table 2: Testing data.

picture	color		texture		class
	white	black	iron	non iron	
1	0.22	0	0	0.9	Cat
2	0	0.42	0.63	0	Chair
3	0	0.17	0	0.14	Cow
4	0.46	0	0	0.37	Clothes

At the beginning, assume two inputs x1, y1∈ [0,1], that x1, y1 are features of images and a weight matrix $w = \{w_{ij}\} \in [0,1]$ and i∈ {1,2}, $j \in \{1,2\}$. An input matrix X($\begin{pmatrix} x1 & y1 \\ \neg x1 & \neg y1 \end{pmatrix}$) is generated then conjunction is applied, following formulas in logic are used:

$$(x1 \wedge_{w_{11}.w_{12}} y1) = (x1 + (1 - w_{11}) - x1(1 - w_{11})) * (y1 + (1 - w_{12}) - y1(1 - w_{12})) \quad (28)$$

$$((\neg x1) \wedge_{w_{21}.w_{12}} y1) = ((1 - x1) + (1 - w_{11}) - (1 - x1)(1 - w_{11})) * (y1 + (1 - w_{12}) - y1(1 - w_{12})) \quad (29)$$

$$((\neg x1) \wedge_{w_{21}.w_{22}} (\neg y1)) = ((1 - x1) + (1 - w_{11}) - (1 - x1)(1 - w_{11})) * ((1 - y1) + (1 - w_{12}) - (1 - y1)(1 - w_{12})) \quad (30)$$

$$(x1 \wedge_{w_{11}.w_{22}} (\neg y1)) = (x1 + (1 - w_{11}) - x1(1 - w_{11})) * ((1 - y1) + (1 - w_{12}) - (1 - y1)(1 - w_{12})) \quad (31)$$

and

$$\neg x1 = 1 - x1 \quad (32)$$

$$\neg y1 = 1 - y1 \quad (33)$$

Conjunction between two elements is according to following picture(1. Orange arrow 2. Green arrow 3. Blue arrow 4. Yellow arrow).

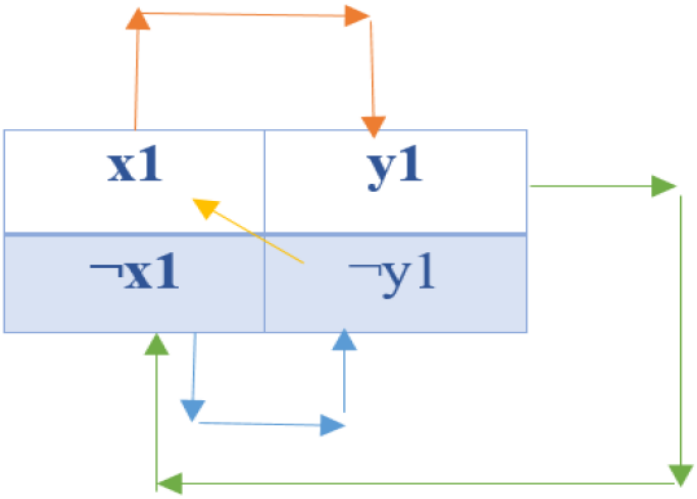

Figure 2: Conjunction between two elements.

Then disjunction is applied between minterms that have been created previously. Because the goal is to find a formula and every formula could be express by normal form and here disjunctive normal form is used.

$$\vee_{w'_{11}.w'_{12}.w'_{21}.w'_{22}} (A.B.C.D)$$

Here, every minterm has a weight of $w'.w' = w'_{ij} \in [0,1]$ and this matrix is generated by learning. The goal is to have an arithmetic form from the formula above. So, following formulas are used (with two inputs x1, y1):

$$(x1 \wedge_{w_{11}.w_{12}} y1) = A \tag{34}$$

$$((\neg x1) \wedge_{w_{21}.w_{12}} y1) = B \tag{35}$$

$$((\neg x1) \wedge_{w_{21}.w_{22}} (\neg y1)) = C \tag{36}$$

$$(x1 \wedge_{w_{11}.w_{22}} (\neg y1)) = D \tag{37}$$

$$A \vee_{w'_{11}.w'_{12}} B = (A * w'_{11}) + (B * w'_{12}) \tag{38}$$

$$\vee_{w'_{11}.w'_{12}.w'_{21}.w'_{22}} (A.B.C.D) = A * w'_{11} + B * w'_{12} + C * w'_{21} + D * w'_{22} \tag{39}$$

6.1. logical rules in CNN

According to formula above, we can use them in CNN. In CNN, formulas below are basic statements of neuron in which output depends on the sum of product of the input with a weight. It is named a score function that maps the raw data to class scores:

$$s(x.w.b) = w * x + b \tag{40}$$

$$f(x.w) = w * x \tag{41}$$

In our work, this formula is developed to $s(X.w.b) = f(X.w) + b$ in CNN(in this example $i \in \{1.2\}$, $j \in \{1.2\}$ and $i' \in \{1.2\}$, $j' \in \{1.2\}$ and $f(X.w)$ is developed to):

$$f(X_{ij}.w_{ij}) = eval(X_{ij}.X_{i'j'}) = \begin{cases} eval(X_{ij}.X_{11}) & \text{if } i = n \text{ and j = n.} \\ eval(X_{ij}.X_{2i}) & \text{if } i = 1 \text{ and j = n.} \\ eval(X_{ij}.X_{ij+1}) & \text{if } i \text{ and j.} \end{cases} \tag{42}$$

According to the 24th rule to 27th one, we have:

$$eval(X_{ij}.X_{11}) = (X_{ij} + (1 - w_{ij}) - X_{ij}(1 - w_{ij})) * (X_{11} + (1 - w_{11}) - X_{11}(1 - w_{11})) \tag{43}$$

$$eval(X_{ij}.X_{2i}) = (X_{ij} + (1 - w_{ij}) - X_{ij}(1 - w_{ij})) * (X_{2i} + (1 - w_{2i}) - X_{2i}(1 - w_{2i})) \tag{44}$$

$$eval(X_{ij}.X_{ij+1}) = (X_{ij} + (1 - w_{ij}) - X_{ij}(1 - w_{ij})) *$$

$$(X_{ij+1} + (1 - w_{ij+1}) - X_{ij+1}(1 - w_{ij+1})) \tag{45}$$

Where X_{ij} belongs to matrix X for example value of X_{11} is x1 and X_{12} is y1 ,... . Also there is $|\{f_{ij}\}| = 2 * N$ that N is the number of input (in this example N=2). In output is used ReLu (The range of ReLu is[0, s1) because in each layer, loss function is reducing):

$$F(s) = max(0.s) \tag{46}$$

Target function is the loss function:

$$L_i = \sum_{j \neq y_i} max(0.s_i - s_{y_j} + 1) \tag{47}$$

$$s_j = f(X.w)_j \tag{48}$$

$$f(X_{ij}.w_{ij}) = eval(X_{ij}.X_{i'j'}) = (X_{ij} + (1 - w_{ij}) - X_{ij}(1 - w_{ij})) * (X_{i'j'} + (1 - w_{i'j'}) - X'_{i'j}(1 - w_{i'j'})) \tag{49}$$

Function $eval(X_{ij}, X_{i'j'})$ has been explained earlier. The proposed method is based on Fig. 3.

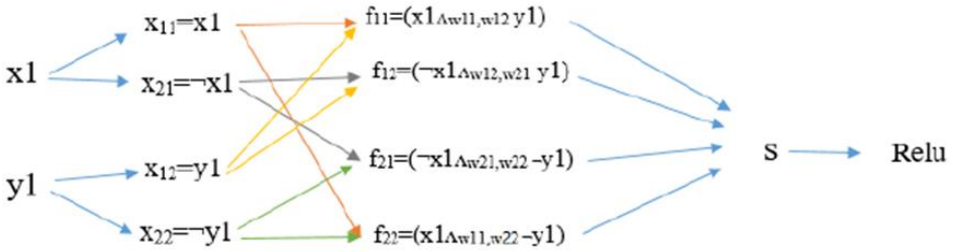

Figure 3: Paper method.

Where the orange arrows have a weight w_{11}, the gray arrows have a weight w_{21}, the yellow ones have a weight w_{12} and the green ones have a weight w_{22}. For example, assume CNN searches a cat and matrix X is generated from two values x1=iron=0.19, y1=white=0.8 (with four classes according to Fig. 4).

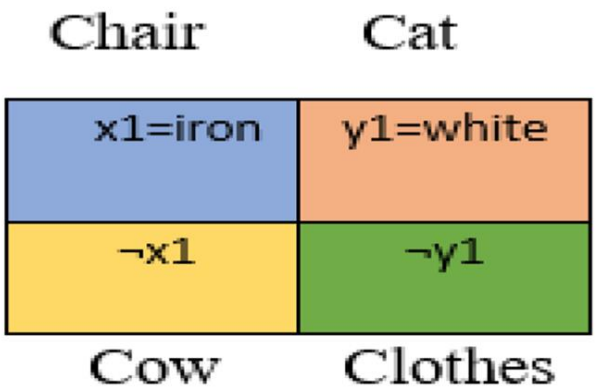

Figure 4: matrix X from two values x1, y1.

The objective is to compare results obtained from CNN and proposed approach. Now consider matrix X that has been generated from 0.19, 0.8, (1-0.19), (1-0.8): $\begin{pmatrix} 0.19 & 0.8 \\ 0.81 & 0.2 \end{pmatrix}$ and matrix w (Weighting matrix values are randomly selected.): $\begin{pmatrix} 0.95 & 0.87 \\ 0.48 & 0.683 \end{pmatrix}$ and b: $\begin{pmatrix} 0.1 & 0.12 \\ 0.08 & 0.01 \end{pmatrix}$. There is $f(X_{ij}.w_{ij}) = w_{ij} * X_{ij}$ in CNN (without our method $i \in \{1.2\}$, $j \in \{1,2\}$) that result is: $\begin{pmatrix} 0.1805 & 0.696 \\ 0.388 & 0.136 \end{pmatrix}$. With considering outputs and b result matrix is $\begin{pmatrix} 0.1805 & 0.696 \\ 0.388 & 0.136 \end{pmatrix} + \begin{pmatrix} 0.1 & 0.12 \\ 0.08 & 0.01 \end{pmatrix} = \begin{pmatrix} 0.2805 & 0.816 \\ 0.468 & 0.137 \end{pmatrix}$. Here, loss function is L=0.359. Relu is:

$$F(f(X.w)+b)=max(0.f(X.w)+b)=\begin{pmatrix}0.2805 & 0.816\\ 0.468 & 0.137\end{pmatrix}$$

result is y1=0.816 in CNN. It means this is a member of the Cat class with feature of white.

Now consider matrix X that has been generated from 0.19, 0.8, (1-0.19), (1-0.8): $\begin{pmatrix}0.19 & 0.8\\ 0.81 & 0.2\end{pmatrix}$ and w: $\begin{pmatrix}0.95 & 0.87\\ 0.48 & 0.683\end{pmatrix}$ and according to our method there are:

$$f(X_{ij}.w_{ij})=eval(X_{ij}.X_{i'j'})=(X_{ij}\wedge_{w_{ij}.w_{i'j'}}X_{i'j'})=(X_{ij}+(1-w_{ij})-X_{ij}(1-w_{ij}))$$

$$*\,(X_{i'j'}+(1-w_{i'j'})-X_{i'j'}(1-w_{i'j'}))$$

$$eval(X_{11}.X_{12})=(X_{11}\wedge_{w_{11}.w_{12}}X_{12})=(X_{11}+(1-w_{11})-X_{11}(1-w_{11}))$$

$$*\,(X_{12}+(1-w_{12})-X_{12}(1-w_{12}))=0.18$$

In fact $f(X.w)$ is: $f(X.w)=\begin{pmatrix}0.18 & 0.75\\ 0.40 & 0.103\end{pmatrix}$, with considering b result is $\begin{pmatrix}0.18 & 0.75\\ 0.40 & 0.103\end{pmatrix}+\begin{pmatrix}0.1 & 0.12\\ 0.08 & 0.01\end{pmatrix}=\begin{pmatrix}0.28 & 0.87\\ 0.48 & 0.113\end{pmatrix}$ because there is this result matrix (with 4 classes (Fig. 5)).

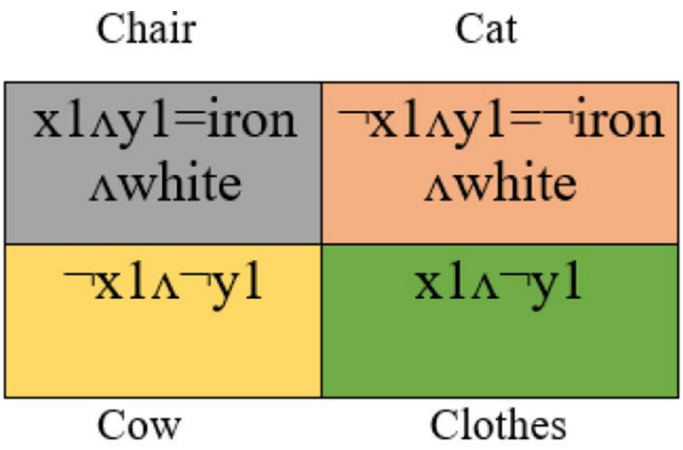

Figure 5: classes of matrix.

and Relu is:

$$F(f(X.w)+b)=max(0.f(X.w)+b)=\begin{pmatrix}0.28 & 0.87\\ 0.48 & 0.113\end{pmatrix}$$

6.2. Training

For training data following equation is used:

$$X^{r+1}=max(0.(f^r(X.w)+b^r)) \tag{50}$$

where X^{r+1} is X in layer r+1. And for training of weighting:

$$w^{r+1}=|w_r+\gamma^r(\nabla_{w^r}Q(Z,w^r))| \tag{51}$$

$$Q(Z.w^r)=l((X_{ij}^r+(1-w_{ij}^r)-X_{ij}^r(1-w_{ij}^r))*(X_{i'j'}^r+(1-w_{i'j'}^r)-X_{i'j'}^r(1-w_{i'j'}^r))) \tag{52}$$

∇ is gradient operator. γ is the learning rate ($\gamma\sim t^{-2}$ and $t>1$) and Z is a output of $f(X.w)$ and l is loss function of $f(X.w)$.

$$Z=f(X.w) \tag{53}$$

$$L_i=\sum_{m\neq y_i}max(0.s_m-s_{y_i}+1) \tag{54}$$

$$l=\frac{1}{N}\sum_{i=1}^{N}L_i \tag{55}$$

$$s_m = f(X.w)_m + b \tag{56}$$

$$f(X_{ij}.w_{ij}) = eval(X_{ij}.X_{i'j'}) = (X_{ij} + (1 - w_{ij}) - X_{ij}(1 - w_{ij}))$$
$$(X_{i'j'} + (1 - w_{i'j'}) - X_{i'j'}(1 - w_{i'j'})) \tag{57}$$

Function $eval(X_{ij}.X_{i'j'})$ has been explained earlier. Here, using the weight of training formula, the weight matrix is $w' = \begin{pmatrix} 0.5695 & 0.357 \\ 0.0925 & 0.33 \end{pmatrix}$ (According to next formula: $|w_r + \gamma^r(\nabla_{w^r} Q(Z.w^r))|$). Then disjunction is applied:

$$\vee_{w'_{11},w'_{12},w'_{21},w'_{22}} (A.B.C.D) = A * w'_{11} + B * w'_{12} + C * w'_{21} + D * w'_{22} = 0.441$$

that result is a cat with white and non iron features.

7. Finding logical formulas from trained CNN

The purpose is to find applied rules in trained CNN. At first, for creating this CNN, Assume two inputs a1, b1 ∈ [0,1] (There are a weight matrix w ∈ [0,1] and a bias matrix b ∈ [0,1] for inputs) and one output and 10 nodes in trained CNN. There are 100 training data and this CNN is according to picture below:

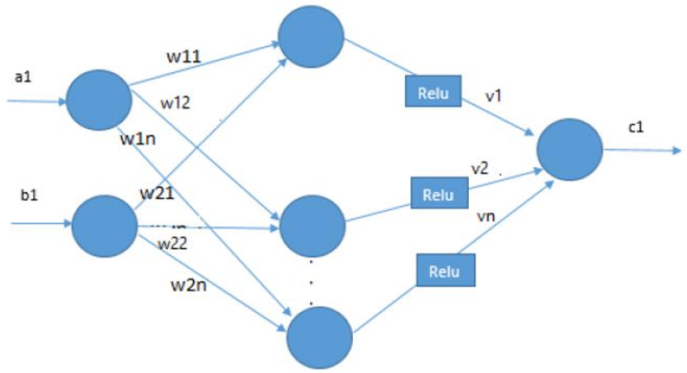

Figure 6: Trained CNN .

The algorithm of training is described in table 3:

Table 3: Algorithm 1.

Algorithm of training:
trainingdata(target):
stepnumber = math.sqrt(numbertraining)
step=1.0/(stepnumber-1.0)
for t in range(numbertraining):
i1 = t // stepnumber
i2 = t mode stepnumber
a1=i1*step
b1 = i2 * step
result[0,t] = a1
result[1,t] = b1
result[2,t] = target(a1,b1)
return result

Here, stepnumber is the steps number in training and numbertraining is the number of training data, target is one of logical rules (OR or AND rules that OR rule as target is considered here). Then SGD optimization is applied because after optimization, error is reduced. Finally, input value is multiplied in the weight and sum with the bias:

$$s(x.w.b) = w * x + b \tag{58}$$

where x is input matrix (it consists of algorithm 1 result and optimization result, in fact is a matrix of a1, b1) and w is a weight matrix (it consists of optimization result) and b is a bias matrix (it consists of optimization result). Then Relu is applied (on output) because CNN is linear, moreover, rules that is used in trained CNN are non linear and the goal is to work without this limitation which Relu do it for us.

$$F(s) = max(0.s) \tag{59}$$

Every result from Relu has a weight ($v_i \in [0.1]$, $i \in \{1.2.3.4\}$) and The result is multiplied by the weight. Now, the output is compared with logical formulas output as following seven logical rules:

$$(a1 \wedge b1) = a1 * b1 \tag{60}$$

$$(\neg a1 \wedge b1) = (1.0 - a1) * b1 \tag{61}$$

$$(a1 \wedge \neg b1) = a1 * (1.0 - b1) \tag{62}$$

$$(\neg a1 \wedge \neg b1) = (1.0 - a1) * (1.0 - b1) \tag{63}$$

$$(a1 \vee b1) = a1 + b1 - a1 * b1 \tag{64}$$

$$\neg a1 = (1.0 - a1) \tag{65}$$

$$\neg b1 = (1.0 - b1) \tag{66}$$

Where OR rule with trained inputs (algorithm 1) is applied separately then the output of trained CNN is compared with OR rule output, this result is reached (blue: OR rule outputs and red: trained CNN outputs (Fig. 7)).

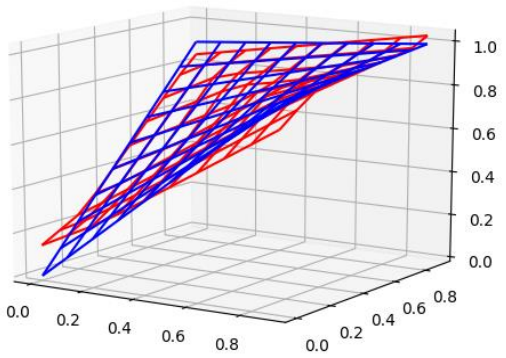

Figure 7: comparing output of CNN and OR rule (blue: OR rule outputs and red: trained CNN outputs).

It is obvious that the result of trained CNN is close to OR rule output. Then we proved this rule is used in trained CNN. Finally, correlation is applied for comparison between AND rules outputs and CNN outputs(Table 4). The least value of results belongs to $(\neg a1 \wedge \neg b1)$=(1.0-a1)*(1.0-b1) since, this logical formula has not been used in LCNN (because as we know $(a1 \vee b1) = (a1 \wedge b1) \vee (\neg a1 \wedge b1) \vee (a1 \wedge \neg b1)$):

Table 4: Correlation between AND rules outputs and CNN output.

Rule	Correlation
$(a1 \wedge b1)$	0.85068792104721

$(\neg a1 \wedge b1)$	0.67625582218170
$(a1,\wedge \neg b1)$	0.69745105504989
$(\neg a1 \wedge \neg b1)$	0.523018956184387

8. Conclusion

In our paper, it is proven that there is a possibility to go from CNN to LCNN and vice versa. First, our work is stared with CNN and applied rules and changed score function with rules then showed that there is a relation between input and output in CNN. in the second part, the work is started with trained CNN and proved that the results of CNN and some rules are almost same, it means there are logical formulas in trained CNN and CNN is created with these rules. For further extension of our work, it would be interesting to capture LCNN with more Performance and extracting more rules from trained CNN.

References

1. Xi, Z., Panoutsos, G., Interpretable Machine Learning: Convolutional Neural Networks with RBF Fuzzy logic Classification Rules. International Conference on Intelligent System, IEEE, Portugal, pp. 448-454 (2018)
2. Gridach, M., Haddad, H., Mulki H.: Churn Identification in Microblogs using Convolutional Neural Networks with Structured Logical Knowledge. Association for Computational Linguistics, Denmark, Denmark, pp. 21-30, (2017)
3. Yhang, B., Xu, X., Li, X., Chen, X., Ye, Y., Wang, Y.: Sentiment analysis through critic learning for optimizing convolutional neural networks with rules. Neurocomputing, Elsevier, 21-30 (2019)
4. Fang, W., Zhang, J., Wang, D., Chen, Z., Li, M.: Entity disambiguation by knowledge and text jointly embedding. Proceedings of the Twentieth SIGNLL Conference on Computational Natural Language Learning, Association for Computational Linguistics, 2016, Germany, pp. 260-269, 10.18653/v1/K16-1026
5. Dai, W.-Z., Xu, Q.-L., Yu, Y., Zhou, Z.-H.: Tunneling neural perception and logic reasoning through abductive learning.Artificial Intelligence, Computer Science, CoRR abs/1802.01173 (2018)
6. Schuhmacher, M., Ponzetto, S.P.: Knowledge-based graph document modeling. Proceedings of the Seventh ACM International conference on Web Search and Data Mining, ACM, USA, pp. 543-552 (2014)
7. Hu, Z., Yang, Z., Salakhutdinov, R., Xing, E.: Deep neural networks with massive learned knowledge. Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Texas, pp. 1670-1679 (2016). 10.18653/v1/D16-1173
8. Alashkar, T., Jiang, S., Wang, S., Fu, Y.: Examples-rules guided deep neural network for makeup recommendation. AAAI, USA, pp. 941-947 (2017)
9. Ruder, S., Ghaffari, P., Breslin, J.G., A hierarchical model of reviews for aspect- based sentiment analysis. Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Texas, pp. 999-1005 (2016). 10.18653/v1/D16-1103
10. Schmitt, I.: Incorporating Weights into a Quantum-Logic-Based Query Language. Quantum-Like Models for Information Retrieval and Decision-Making. Springer, Switzerland, 129–139 (2019)
11. Kumar Saha, S., Schmitt, I.: Non-TI Clustering in the Context of Social Networks. The 2nd International Workshop on Web Search and Data Mining (WSDM)April 6 - 9, Warsaw, Poland, Science Direct, 1186-1191(2020).