1.

, , . , (computer-aided diagnosis) (image segmentation)(image registration)(image fusion)(image-guided therapy)(image annotation)(image database retrieval), , .:[?], [?], [?], .

, . , [?]. . , CT. :

2.

2.1.

(neurons) . , . 6, x_i, (bias). (1).w_i , b , $f(.)$ (nonlinear activation function).

$$output = f(\sum_{i=1}^{3} w_i x_i + b) \tag{1}$$

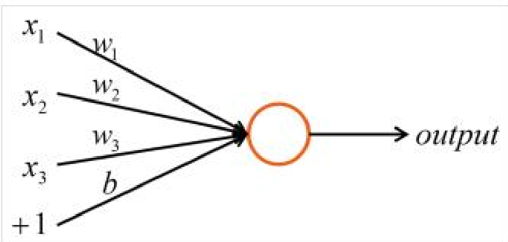

Figura 1.

, x_i w_i. b_i. $f(.)$. : S(Logistic Sigmoid Function), (Hyperbolic Tangent Function), (Rectified Linear Unit, ReLU)(Leaky ReLU). 2.

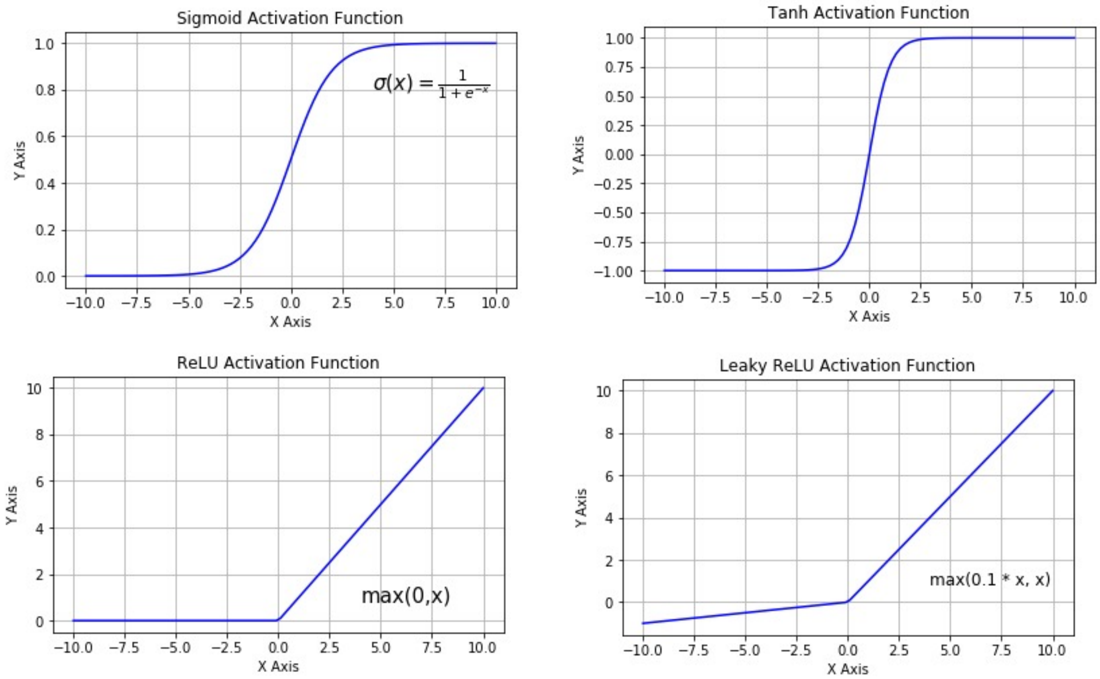

Figura 2. : S(Logistic Sigmoid Function), (Hyperbolic Tangent Function), (Rectified Linear Unit, ReLU)(Leaky ReLU)

(2) - (5)

S(Logistic Sigmoid Function):

(2)

(Hyperbolic Tangent Function):

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{3}$$

(Rectified Linear Unit, ReLU)

$$f(x) = max(0, x) \tag{4}$$

(Leaky ReLU)

$$f(x) = \begin{cases} x & if\ x > 0 \\ 0.01x & otherwise \end{cases} \tag{5}$$

2.2.

(1) +1+1(unit)

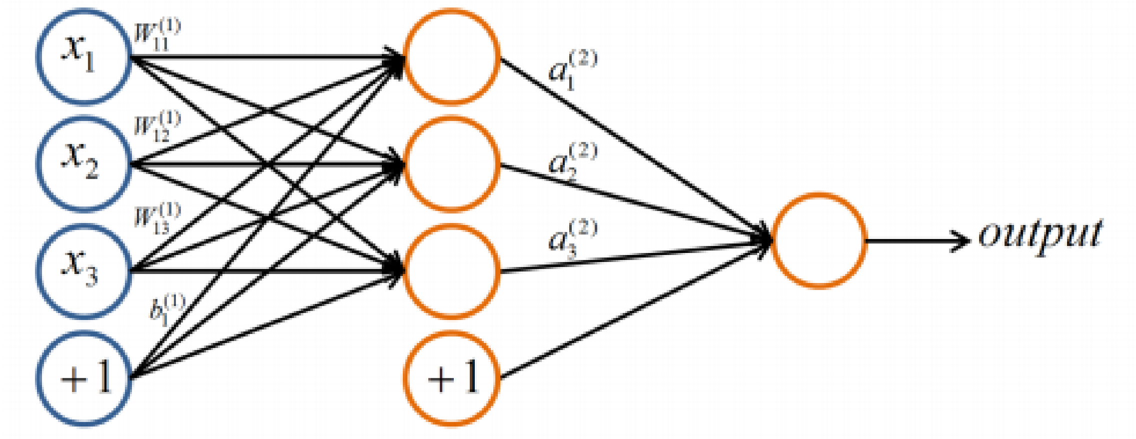

Figura 3. (bias)+1$a_i^{(l)} il$

iL_i, $n_L = 3$. $L_1 L_2$, L_3, $(W, B) = (W^{(1)}, b^{(1)}, W^{(2)}, b^{(2)})$, $b_i^{(l)}$i$W_{ij}^{(l)}$ li$l+1$j$W^{(1)} \in \mathbb{R}^{3\times3}$, $W^{(2)} \in \mathbb{R}^{1\times3}$, $b^{(1)} \in \mathbb{R}^3$ and $b^{(2)} \in \mathbb{R}^1$.

$$a_1^{(2)} = f(W_{11}^{(1)} x_1 + W_{12}^{(1)} x_2 + W_{13}^{(1)} x_3 + b_1^{(1)}) \tag{6}$$
$$a_2^{(2)} = f(W_{21}^{(1)} x_1 + W_{22}^{(1)} x_1 + W_{23}^{(1)} x_1 + b_2^{(1)}) \tag{7}$$
$$a_3^{(2)} = f(W_{31}^{(1)} x_1 + W_{32}^{(1)} x_1 + W_{33}^{(1)} x_1 + b_3^{(1)}) \tag{8}$$

a_i^l l$f(.)$ll+1

$$output = a_1^{(3)} = \sigma(W_{11}^{(2)} a_1^{(2)} + W_{12}^{(2)} a_2^{(2)} + W_{13}^{(2)} a_3^{(2)} + b_1^{(2)}) \tag{9}$$

σ S(Logistic Sigmoid Function), Sigmoid (forward propagation) . $y\{0, 1\}$n$0, 1, ..., n$

2.3.

(Back Propagation Gradient Decent)O_k, $\sigma^i(k), i = 1, 2, ..., n.W$:

$$W(k+1) = W(k) - \lambda \frac{\partial E}{\partial w} \tag{10}$$

:

$$E = \frac{1}{2} \sum_{k=1}^{n} (t_k - O_k)^2 \tag{11}$$

t_kk.

:

:

$$\frac{\partial E}{\partial w_{jk}} = -\sum_k (t_k - O_k) \frac{O_k}{\partial w_{jk}} \tag{12}$$

:

$$\frac{\partial E}{\partial w_{ij}} = -\sum_k (t_k - O_k) \frac{O_k}{\partial w_{ij}} \tag{13}$$

:

$$\frac{\partial E}{\partial w_l} = -\sum_k (t_k - O_k)\frac{O_k}{\partial w_l} = -\sum_k (t_k - O_k)\frac{O_k}{\partial \sigma_k}\frac{\partial \sigma_k}{\partial w_l} \tag{14}$$

1. .

Algorithm 1:

:

(**x**,**y**),**w**, (V, E), **V**, **E**

$\sigma : \mathbb{R} \to \mathbb{R}$

:

$\mathbf{V}_0, ..., \mathbf{V}_T$, $\mathbf{V}_t = \{v_{t,1}, ..., v_{t,k_t}\}$

tt+1($(v_{t,j}, v_{t+1,i})$) $W_t, i, j,\ (v_{t,j}, v_{t+1,i} \notin E), W_{t,i,j} = 0$

:

$\mathbf{O}_0 = \mathbf{x}$

for $t = 1, ..., T$ **do**

  for $i = 1, ..., k_t$ **do**

    $a_{t,i} = \sum_{j=1}^{k_{t-1}} W_{t-1,i,j} O_{t-1,j}$

    $O_{t,i} = \sigma(a_{t,i})$

  end

end

:

$\delta_T = \mathbf{O}_T - \mathbf{y}$

for $t = T-1, T-2, ..., 1$ **do**

  for $i = 1, ..., k_t$ **do**

    $\delta_{t,i} = \sum_{j=1}^{k_{t+1}} W_{t,i,j}\delta_{t+1,j}\sigma'(a_{t+1,j})$

  end

end

:

$(v_{t-1,j}, v_{t,i}) \in E$

$\delta_{t,i}\sigma'(a_{t,i})O_{t-1,j}$

1. .

3. CT

3.1.

, CTCTCTCT

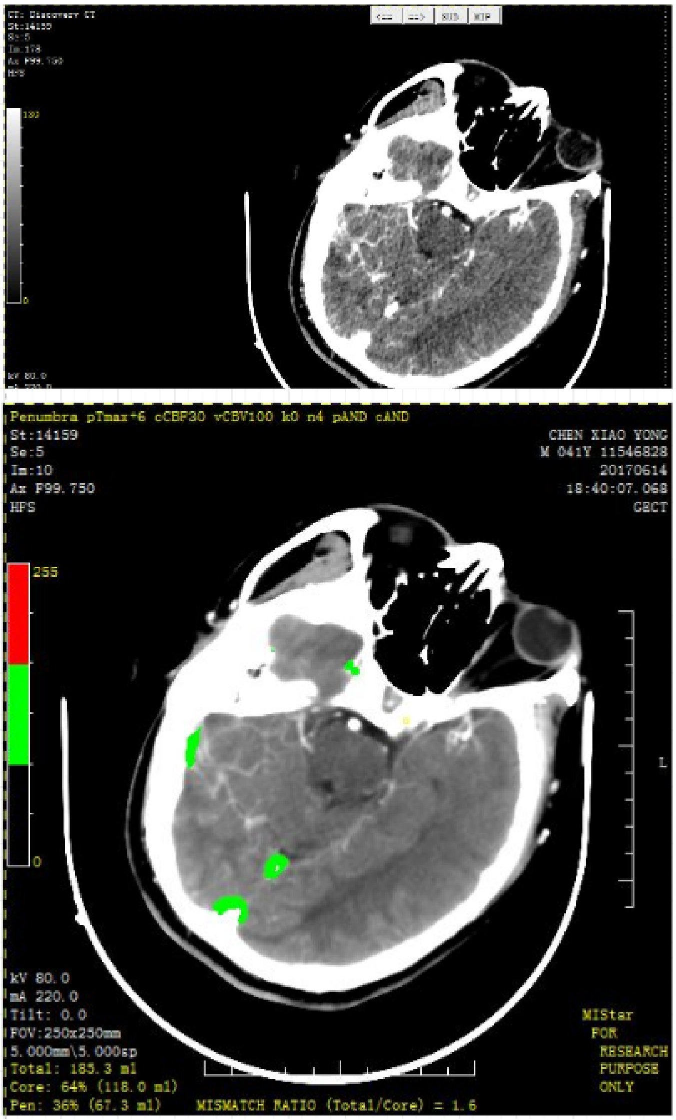

Figura 4. CTCT

(CNN): (Convolutions) , (Subsampling)(fully connected). , , , 5,,1, , , 0, . 6, , ()..

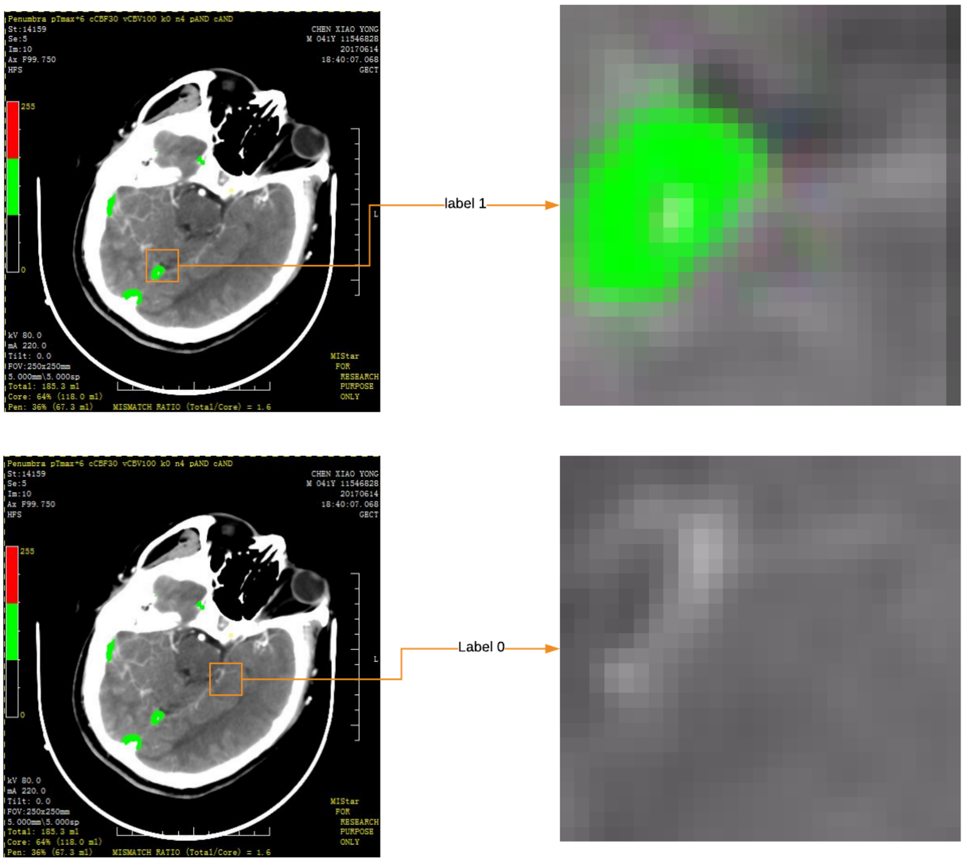

Figura 5. CT: CT422 $\times$ 733, , ,, ,1, , , 0, .

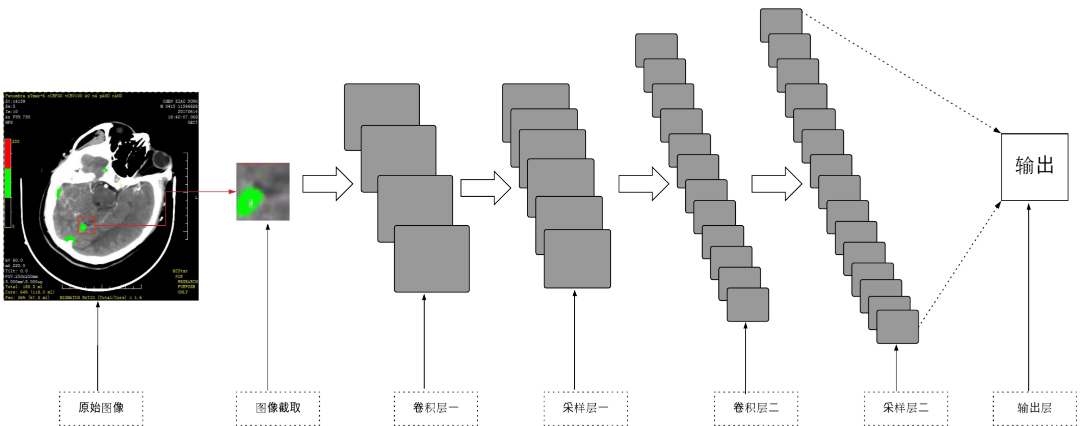

Figura 6. : , ()

3.2.

, Google Tensorflow, KerasOpenCV. KerasTheanoTensorflowPython, . Python 2.73.5KerasGPUCPU.Keras, :

- .
- keras,
- Keras

KerasPython .

Listing 1. Keras example

```
def getModel():
    model=Sequential()

    # CNN 1
    model.add(Conv2D(64, kernel_size=(3, 3),activation='relu', input_sha
    model.add(MaxPooling2D(pool_size=(3, 3), strides=(2, 2)))
    model.add(Dropout(0.2))

    # CNN 2
    model.add(Conv2D(128, kernel_size=(3, 3), activation='relu' ))
    model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
    model.add(Dropout(0.2))

    # CNN 3
    model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
    model.add(Dropout(0.2))

    #CNN 4
    model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
    model.add(Dropout(0.2))

    model.add(Flatten())

    #Dense 1
    model.add(Dense(512, activation='relu'))
    model.add(Dropout(0.2))

    #Dense 2
    model.add(Dense(256, activation='relu'))
    model.add(Dropout(0.2))

    # Output
    model.add(Dense(1, activation="sigmoid"))

    optimizer = Adam(lr=0.001, decay=0.0)
    model.compile(loss='binary_crossentropy', optimizer=optimizer, metr

    return model
```

3.3.

, , , , :. , , .7, ,, 100, . : (Rotation), (transportation), (Shear) (Brightness).

Figura 7.

4.

, , , , , .
https://ani.stat.fsu.edu/ jinfeng/

:. .
:https://jianwang2018.github.io/jianwang.github.io/