$$w' = w - \epsilon \frac{\partial L}{\partial w}$$

ReLU

An introduction of Deep Learning and LSTM

A brief introduction of Deep Learning

Don’t be afraid

- Deep learning is just a neural network!
- In practice, if the number of layers in a neural network is more than four, it is called deep learning (neural network).

- The term, deep learning, has been used from Year 2000. Before that, it was difficult to make a neural network with more than four layers. Why?

Gradient descent

- Update rules (gradient descent) of w and v are

$$w' = w - \epsilon \frac{\partial}{\partial w} R$$

$$v' = v - \epsilon \frac{\partial}{\partial v} R$$

Vanishing gradient problem

$\sigma(1-\sigma) \leq \frac{1}{4}$

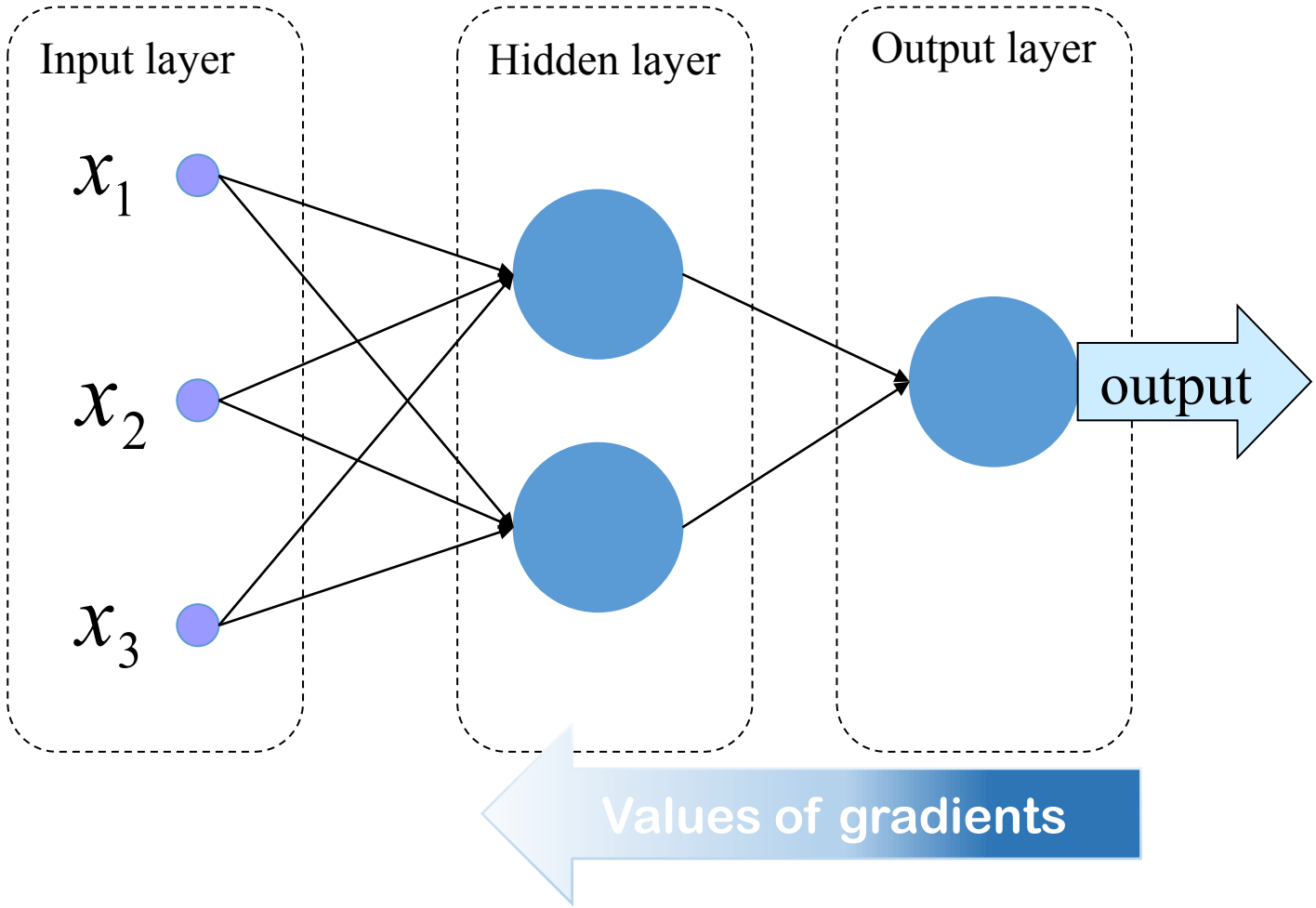
- As an activation function, it has been common to use a sigmoid function.
- In order to train the network, it has been also common to use back propagation because of its good performance.
- As we discussed a neural network,

$$z(x) = \sigma(v\,\sigma(wx - b) - d),$$

the gradient of its quadratic loss function $\frac{\partial}{\partial w}R$ is much less than $\frac{\partial}{\partial v}R$.

- Same situation for a neural network hinder update the value of weights in the layers closed to input layer.
  - Neural networks cannot be trained well ;(

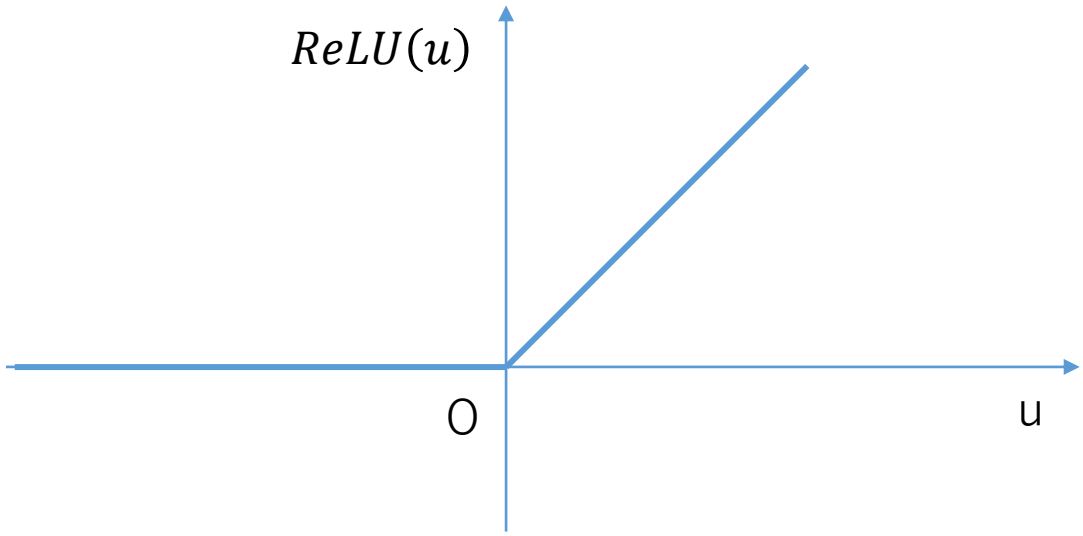
Vanishing gradient problem



ReLU

- ReLU is a new activation function, which does not cause vanishing gradient problem.

$$ReLU(u) = \begin{cases} u & (u \geq 0) \\ 0 & (u < 0) \end{cases}$$



Exercise

- Consider the case two neural networks are given as

$$z_1(x) = \sigma\big(v\,\sigma(wx)\big)$$
$$z_2(x) = ReLU\big(v\,ReLU(wx)\big)$$

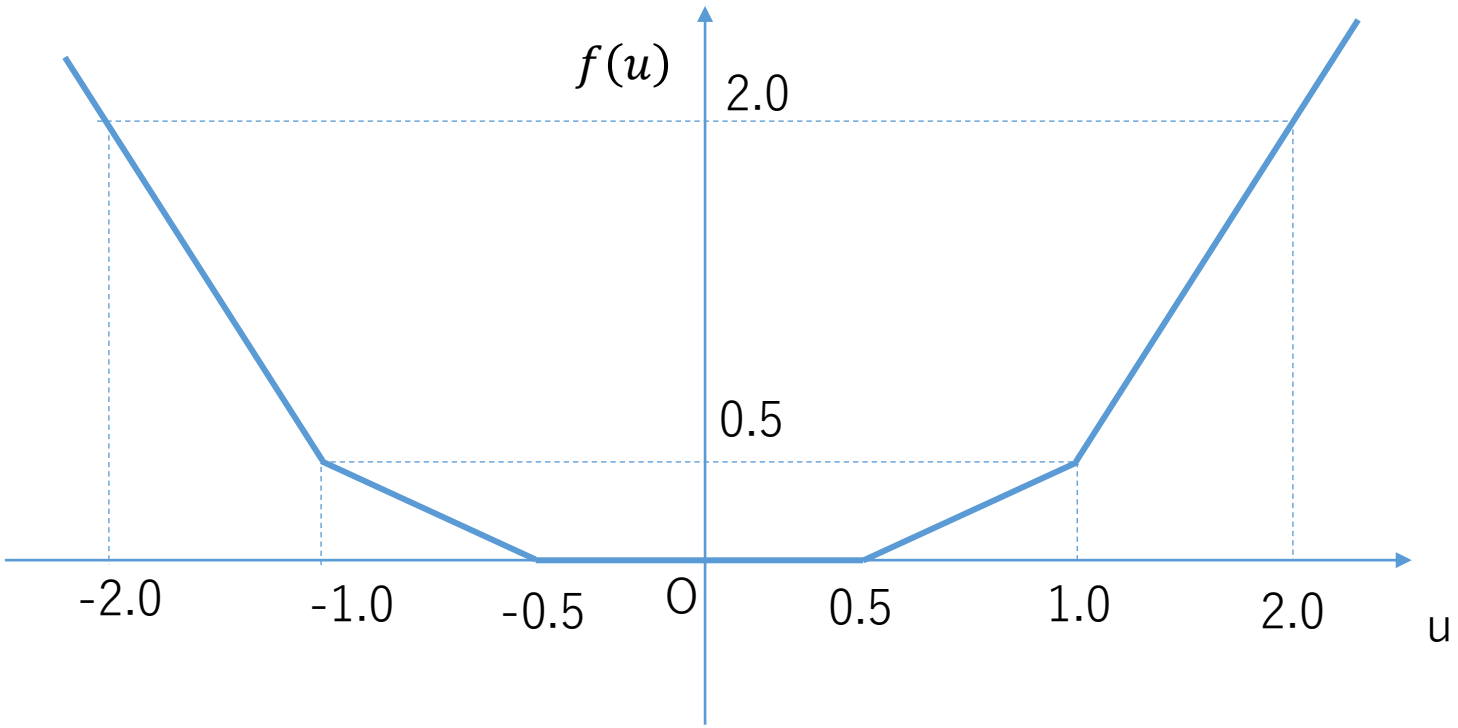
  and $R_i(v,w) = \frac{1}{2}(z_i(x) - 0)^2$.

- For positive v, w and $x(< 1)$, which is larger, $\frac{\frac{\partial R_1}{\partial w}}{\frac{\partial R_1}{\partial v}}$ or $\frac{\frac{\partial R_2}{\partial w}}{\frac{\partial R_2}{\partial v}}$?

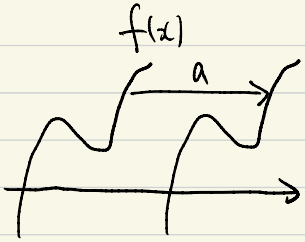
Exercise

$$\sum_i \gamma_i \, \mathrm{ReLU}(\alpha_i u + \theta_i)$$

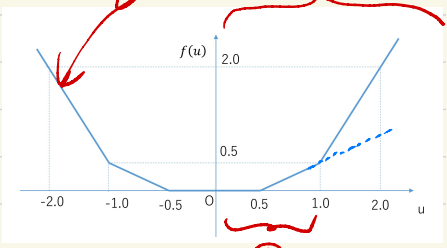
- The following graph is for $f(u)$, which is a non-linear function. Express $f(u)$ by combining four ReLU functions.



Hint!



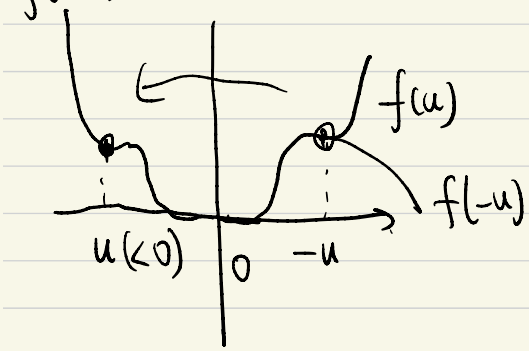
$f(u) \approx \frac{1}{2}u^2$



① ReLU(u − 0.5)

② ReLU(u − 0.5) + 0.5 × ReLU(u − 1.0)

u = 2.0 → 1.5 + u = 2.0 → 0.5 × 1.0 = 2.0



③ ReLU(u − 0.5) + 0.5 × ReLU(u − 1.0)
+ ReLU(−u − 0.5) + 0.5 × ReLU(−u − 1.0)

Using ReLU, we got large NNs

- Alexnet, which makes DL famous, was developed in 2012 and had 5 CNN layers and 3 fully-connected layers.

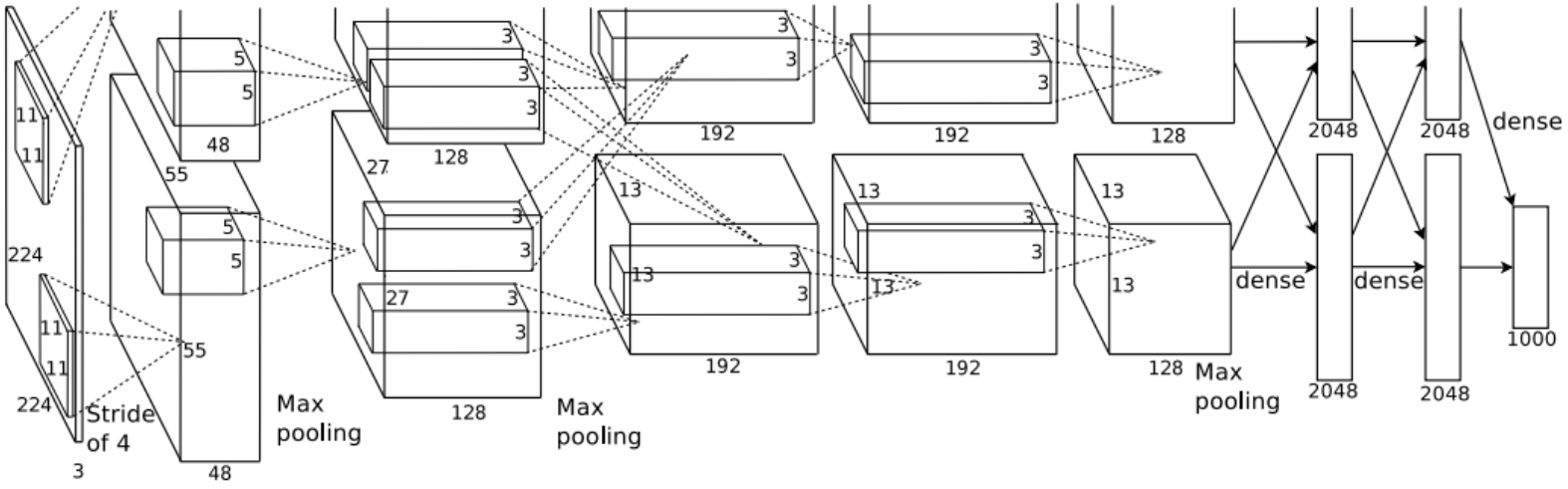


Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input is 150,528-dimensional, and the number of neurons in the network's remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

Recent DLNN with more layers

- The champion network for a image recognition contest (ILSVRC 2015), ResNet, had 152 layers.

34 layers' version

$x + N(x)$

https://arxiv.org/abs/1512.03385

Why “deep” is good?

- We can realize a complex function as a combination of simple functions.
  - Remember that functions can be approximated by a combination of ReLU functions.
  - Consider we have neural networks approximately represent following functions.

$$f_1(x) \simeq e^x$$

$$f_2(x) \simeq \sin x$$

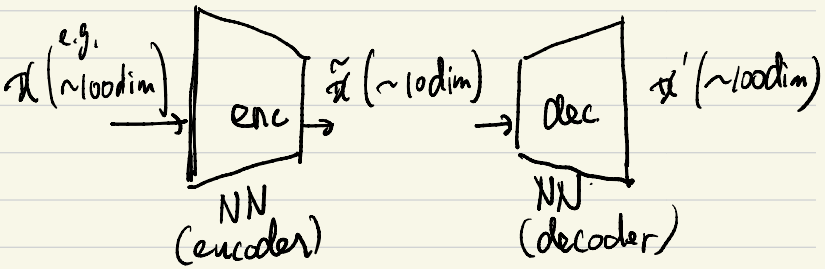
$$f_3(x) \simeq -x^2$$

$$f_4(x) \simeq \log x$$

Then, a deep network $f_1\left(f_2\left(f_3\left(f_4(x)\right)\right)\right)$ approximately express a complex function $e^{\sin(-(\log x)^2)}$

Auto encoder ... dimension reduction

$$D \longrightarrow d \ (\ll D)$$



$$\tilde{x} = enc(x) \qquad x' = dec(\tilde{x})$$

$$L = \sum_i \| dec(enc(x_i)) - x_i \|^2$$

Example

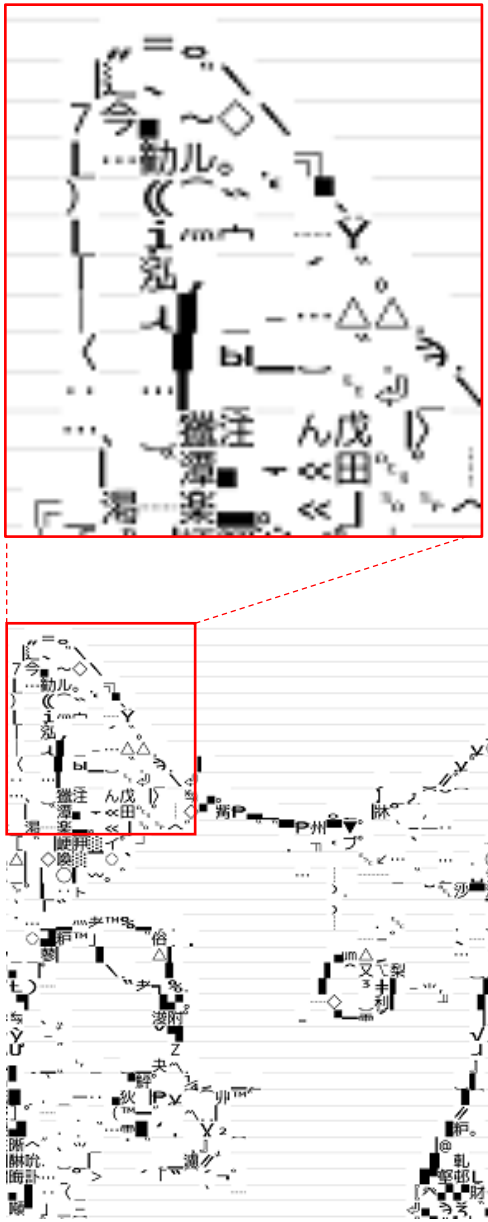
An Autoencoder Based ASCII Art Generator

MASAOMI KIMURA
Shibaura Institute of Technology (SIT), JAPAN

MOHAMMAD IQBAL, IMAM MUKHLASH
Sepuluh Nopember Institute of Technology (ITS), INDONESIA

Background

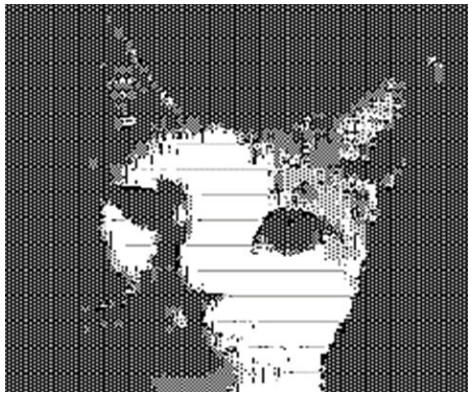
- ASCII art (AA) is the way to realize a picture with letters.
  - Alphabets, numbers and symbols
- Web communities usually use Japanese fonts with
  - Kanji characters(今, 田)
  - Hiragana, Katakana(ル,ん)
  - More complex symbols (△, ﾍﾞ)
- AA is used under the environment that accepts only characters such as BBS in WWW



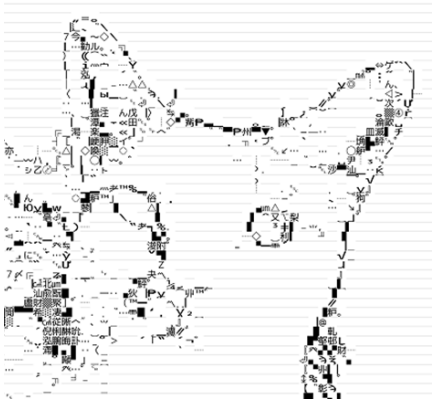
Two classes of ASCII art

- Tone-based ASCII art
  - A portion of shade is expressed with a character depending on brightness
- Structure-based ASCII art
  - A portion of the outline is expressed with a character

Original image



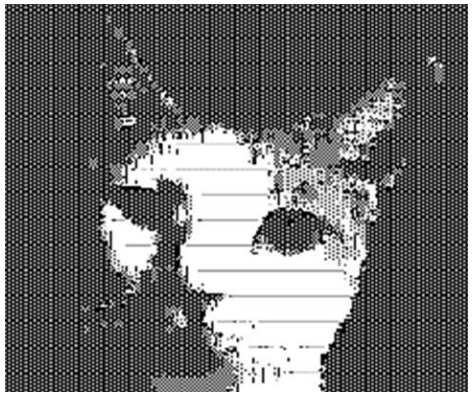
Tone-based AA

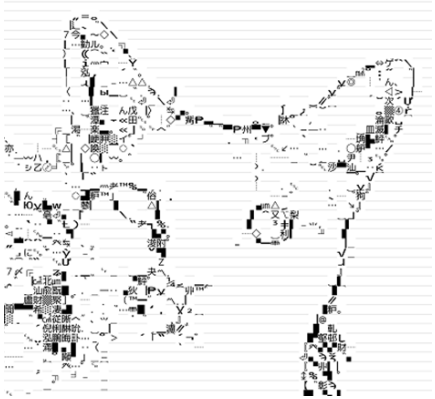


Structure-based AA

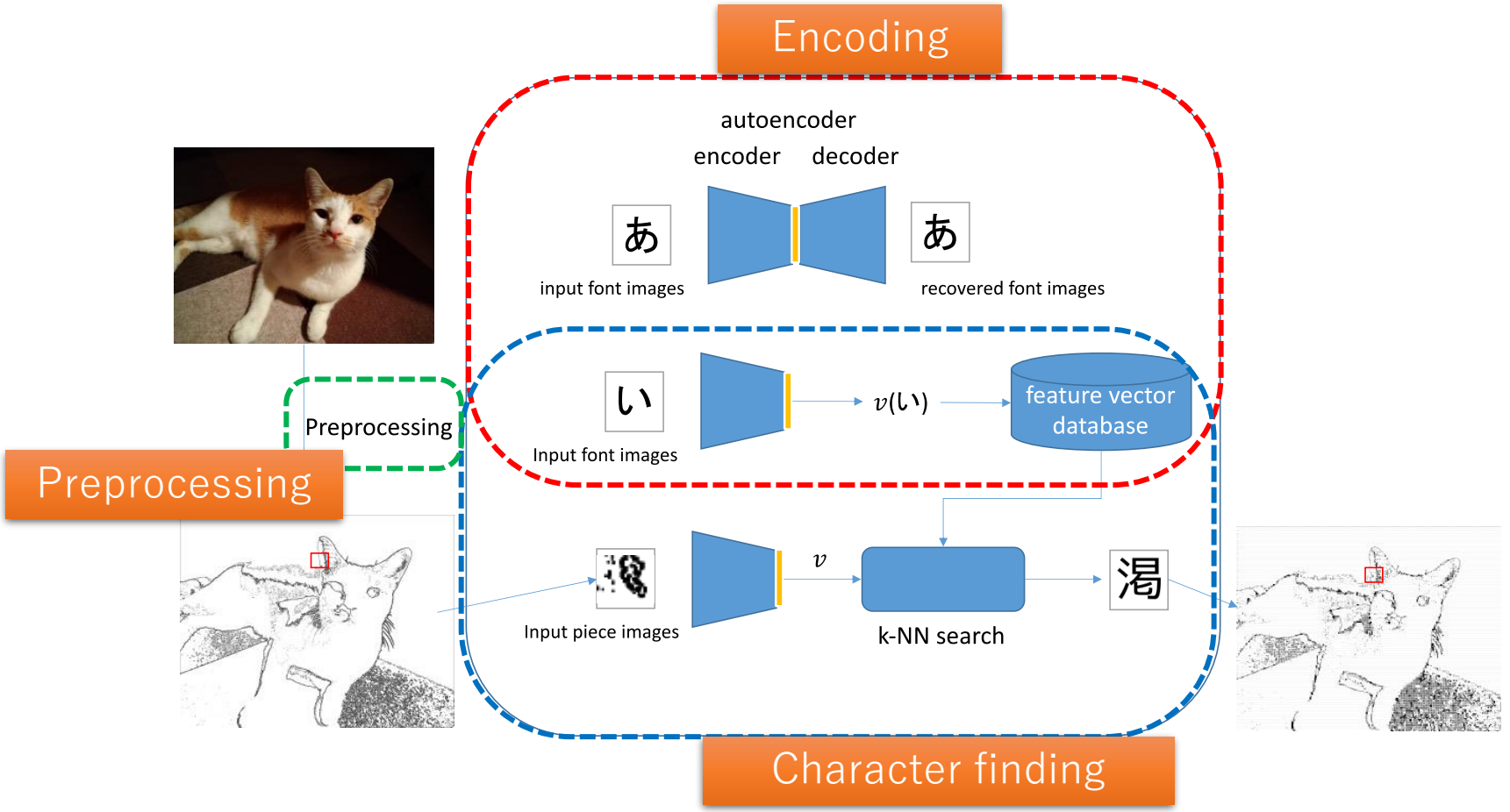
Objective

We propose a model to generate
both tone-based and structure-based ASCII art images





The proposed method
- Overview

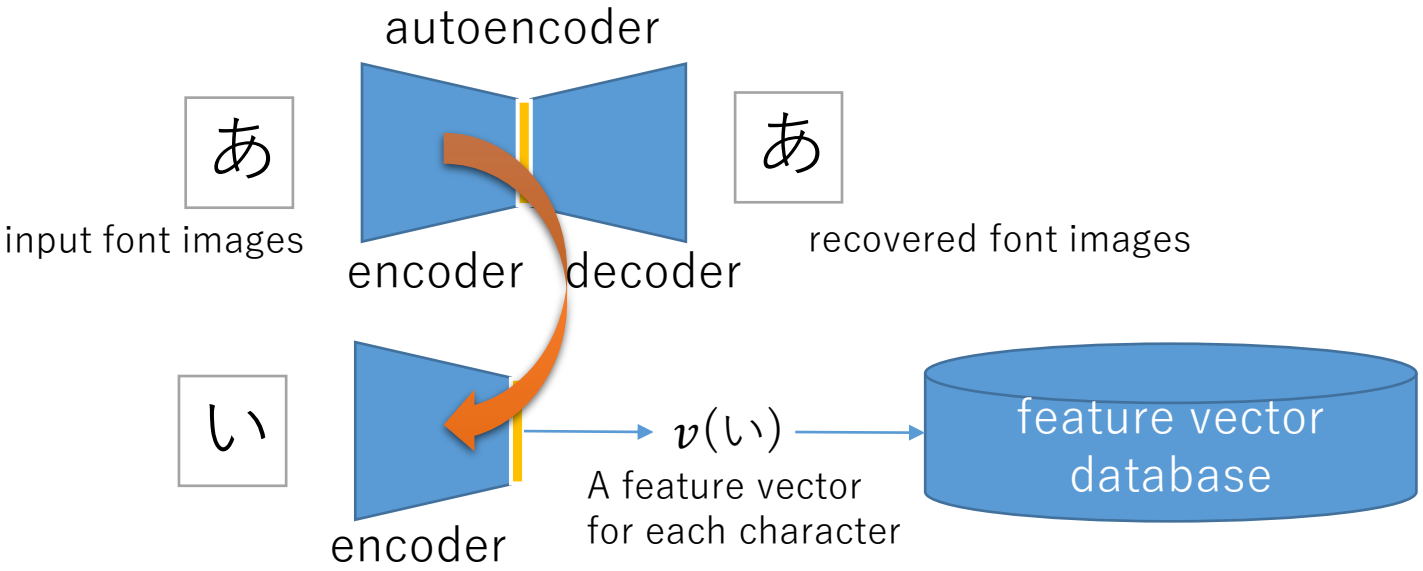


The proposed method
- Preprocessing

1. Grayscale conversion of the target image
2. Increasing its contrast and brightness
3. Magnifying the image
   - to fit the size of characteristic patterns to characters
4. Edge detection (for structure-based AA)

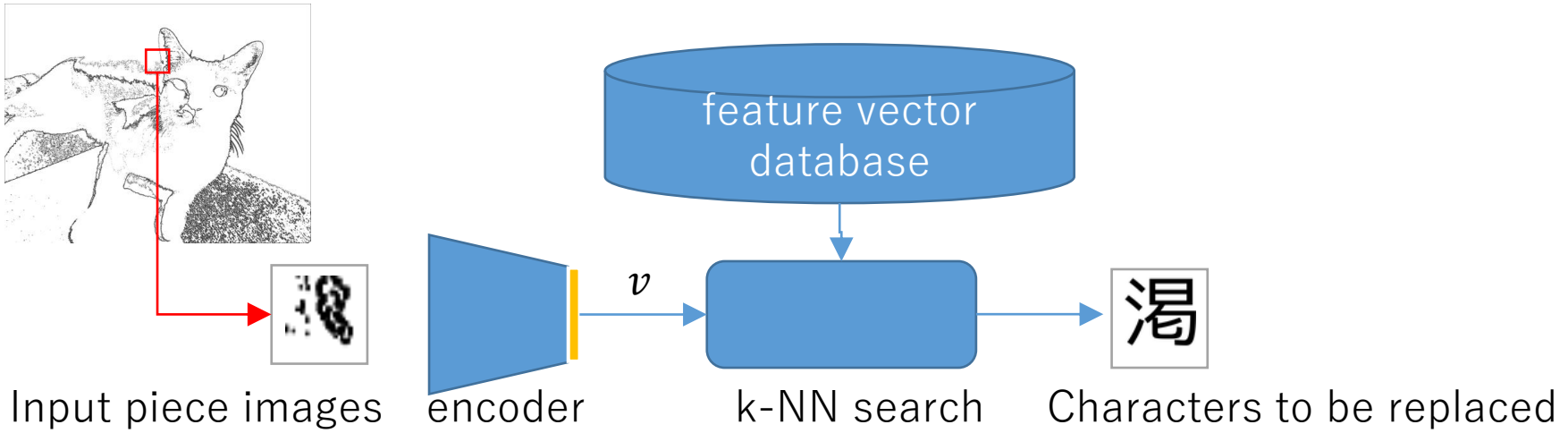
The proposed method
- Encoding using an autoencoder

- An autoencoder is used to extract features and reduce dimensions.
- The encoder part is used to get a feature vector, which is stored to the database.



The proposed method
- Finding a character to be replaced

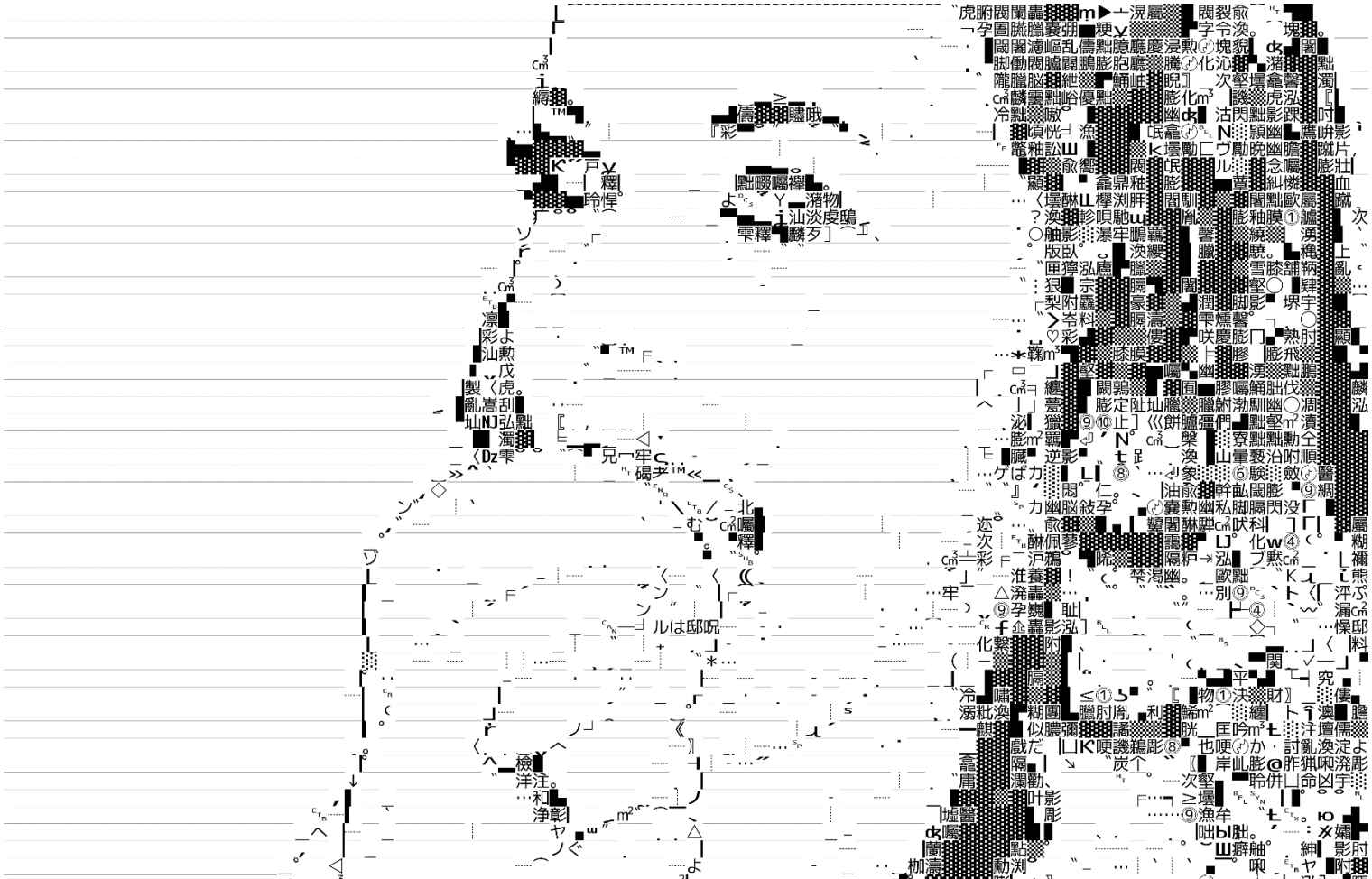
- The target image is separated into pieces of square images
- Again, the encoder is used to convert the piece image into a feature vector.
- We find a character associated to the feature vectors closest to the obtained vector out of the feature vector database.
- The character is pasted to the image at the location of the original piece image.



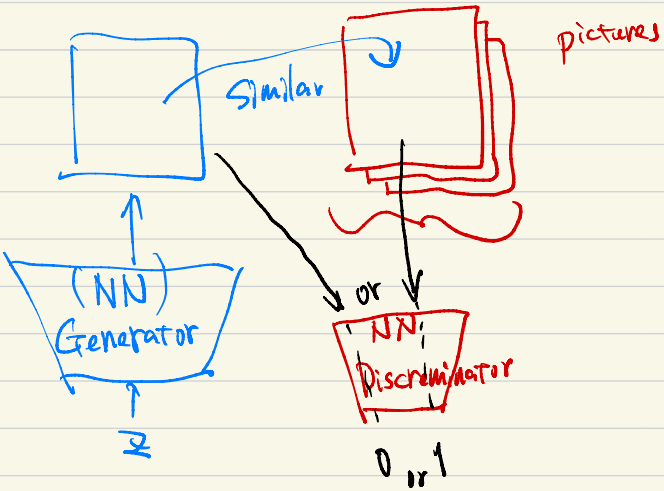
Original image

The generated tone-based AA

The generated structure-based AA



GAN



LSTM

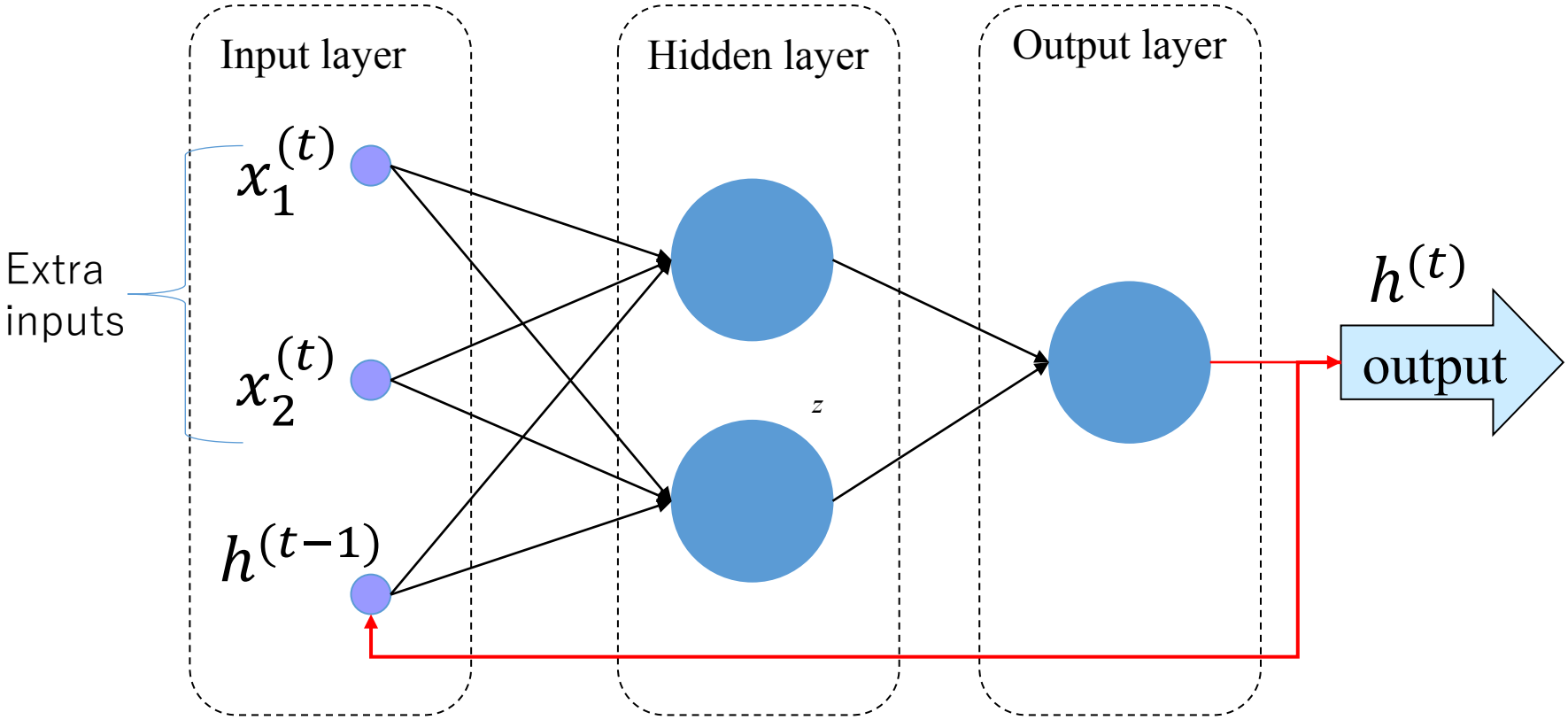
Time series data

- A series of ordered data
  - Can be express as an array of vectors $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \cdots, \boldsymbol{x}^{(t)}, \cdots$
- What is an example?

Date	8/7	8/8	8/9	8/10
Temp	38	36	37	35
Humidity	40%	70%	50%	80%
Weather	Fine	Cloudy	Fine	Rainy

Recurrent Neural Network

- We can make a model which depends not only on an input but also on information at a previous timing.

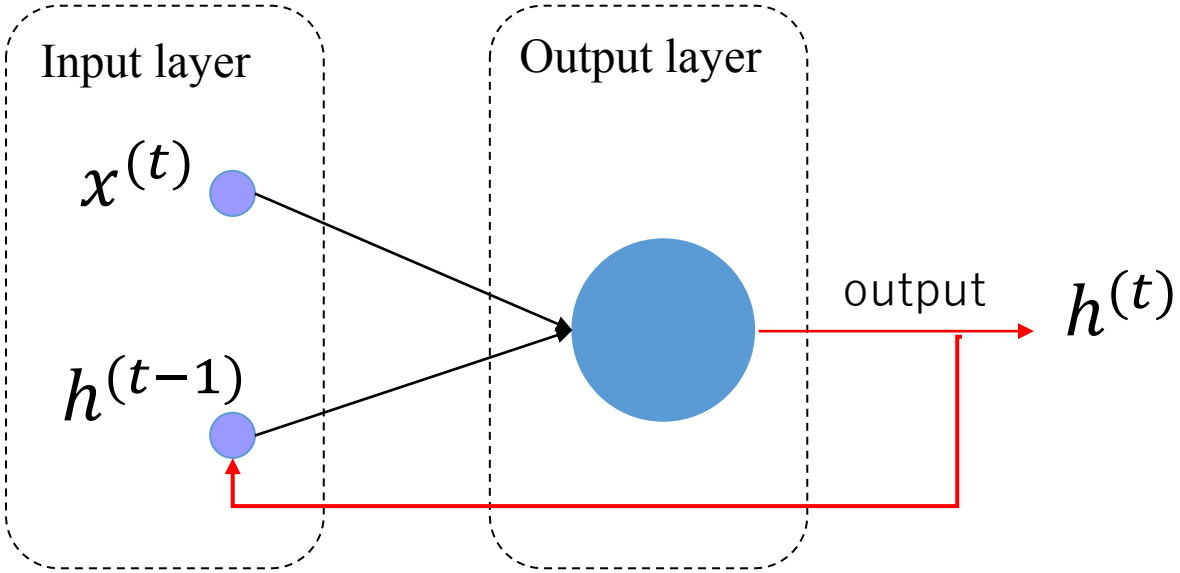


The most simple model is …

- Let's discuss a simple model

$$h^{(t)} = \tanh\left(w_h h^{(t-1)} + w_x x^{(t)} - \theta\right)$$

- In this model, two weights w_x and w_h are constants and are independent from $x^{(t)}$ and $h^{(t-1)}$.



A core idea of the model

- The structure is

$$h^{(t)} = \tanh\left(w_h h^{(t-1)} + w_x x^{(t)} - \theta\right)$$

New output　　Stored old data　　New input

- We can simplify it to

$$h^{(t)} = w_h h^{(t-1)} + w_x x^{(t)}$$

Contribution control of stored data and new input

- Originally, the weights are constants. If we extend them to time-dependent functions,

$$h^{(t)} = w_h(t)h^{(t-1)} + w_x(t)x^{(t)},$$

  we can control the contribution of $h^{(t-1)}$ and $x^{(t)}$.

- [Exercise] If $w_h(t)$ and $w_x(t)$ are positive and $w_h(t) + w_x(t)$=1, what should we interpret the role of the weights?

The idea

- In order to restore non-linearity, we modify the formulation as

$$C^{(t)} = f(t)C^{(t-1)} + i(t)x^{(t)}$$
$$h^{(t)} = o(t)\tanh\left(C^{(t)}\right),$$

  where $0 < f(t) < 1$ and $0 < i(t) < 1$.
- $C^{(t)}$ plays a role of a memory.
- $f(t)$ is called a “forget gate”, which reduces the contribution of the memory $C^{(t)}$.
- $i(t)$ is called an “input gate”, which reduces the contribution of the input $x^{(t)}$.
- $o(t)$ is called an “output gate”, which reduces the output.

General formula of LSTM

$$\boldsymbol{f} = \sigma\left(W_h^f \boldsymbol{h}^{(t-1)} + W_x^f \boldsymbol{x} - \theta^f\right)$$
$$\boldsymbol{i} = \sigma\left(W_h^i \boldsymbol{h}^{(t-1)} + W_x^i \boldsymbol{x} - \theta^i\right)$$
$$\boldsymbol{o} = \sigma\left(W_h^o \boldsymbol{h}^{(t-1)} + W_x^o \boldsymbol{x} - \theta^o\right)$$
$$\widetilde{\boldsymbol{C}} = \sigma\left(W_h^C \boldsymbol{h}^{(t-1)} + W_x^C \boldsymbol{x} - \theta^C\right)$$

$$\boldsymbol{C}^{(t)} = \boldsymbol{f} \odot \boldsymbol{C}^{(t-1)} + \boldsymbol{i} \odot \widetilde{\boldsymbol{C}}$$
$$\boldsymbol{h}^{(t)} = \boldsymbol{o} \odot \tanh\left(C^{(t)}\right)$$

Note : The symbol ⊙ denotes Hadamard product.

Hadamard product

- We have two vectors: $\boldsymbol{v}_1 = \begin{pmatrix} a_1 \\ b_1 \end{pmatrix}$ and $\boldsymbol{v}_2 = \begin{pmatrix} a_2 \\ b_2 \end{pmatrix}$.
- Hadamard product is a element wise product:

$$\boldsymbol{v}_1 \odot \boldsymbol{v}_2 = \begin{pmatrix} a_1 a_2 \\ b_1 b_2 \end{pmatrix}$$

- An example

$$\begin{pmatrix} 2 \\ 3 \end{pmatrix} \odot \begin{pmatrix} 5 \\ 3 \end{pmatrix} = \begin{pmatrix} 10 \\ 9 \end{pmatrix}$$

GRU

- Gated Recurrent Unit
- The purpose of GRU is similar to LSTM: predictions based on time series training data

Update gate

$$\boldsymbol{z} = \sigma\left(W_h^z \boldsymbol{h}^{(t-1)} + W_x^z \boldsymbol{x} - \theta^z\right)$$

Reset gate

$$\boldsymbol{r} = \sigma\left(W_h^r \boldsymbol{h}^{(t-1)} + W_x^r \boldsymbol{x} - \theta^r\right)$$

Memory update

$$\widetilde{\boldsymbol{h}}^{(t)} = \tan\left(W_h^h (\boldsymbol{r} \odot \boldsymbol{h}^{(t-1)}) + W_x^h \boldsymbol{x} - \theta^h\right)$$

$$\boldsymbol{h}^{(t)} = (1 - \boldsymbol{z}) \odot \boldsymbol{h}^{(t-1)} + \boldsymbol{z} \odot \widetilde{\boldsymbol{h}}$$