Text file to create
We need to set red words
The blue part can be omitted

■ Neural network setting「NET.txt」

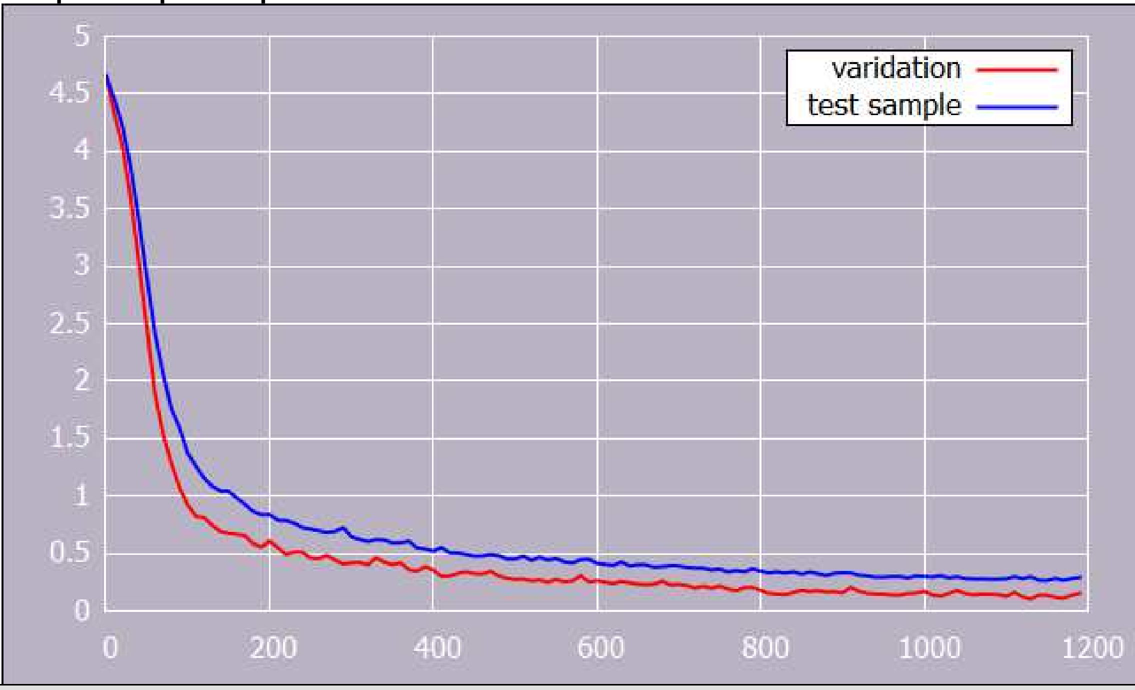
CrossEntropy	**Square** or **CrossEntropy**
BATCH_SIZE **50**	Mini batch size
EPOCH **10**	Epoch number
LAMBDA **0.000000**	Load decay (weight decay)
EPS **0.001000**	Learning rate
OPTIMIZER **Adam**	Optimizer
VALIDATION_NUM **100**	Number of Validation data
ERROR_PLOT_STEP **10**	Graph data output interval(gnuplot format)
TEST_SAMPLE **10**	Number of test data samples
ACCURACY_RATE_PLOT **1**	Accuracy plotting
	※optimizer solver is listed on the final page

~~* Leave the setting of LAMBDA to 0 (there is a problem)~~

ERROR_PLOT_STEP

The loss value is output to '**error_loss.dat**' at the interval specified at the time of learning. **gnuplot** can display graphs in real time.

Gnuplot script example




```
set border lc rgb "white"
set grid lc rgb "white" lt 2
set key opaque box
set object 1 rect behind from screen 0,0 to screen 1,1 fc rgb "#B8B2C3" fillstyle solid

# smooth [unique, csplines, acsplines, bezier, sbezier]

plot 'error_loss.dat' using 1:2    t "varidation"    with lines linewidth 2 linecolor rgbcolor "red"
replot 'error_loss.dat' using 1:3      t "test sample" with lines linewidth 2 linecolor rgbcolor "blue"

#replot 'error_loss.dat' using 1:2 t "varidation"    with lines linewidth 1 linecolor rgbcolor "red"
#replot 'error_loss.dat' using 1:3 t "test sample" with lines linewidth 1 linecolor rgbcolor "blue"

pause 10
reread
```

ACCURACY_RATE_PLOT

Accuracy is output to ‘**accuracy_rate.dat**’ at the interval specified at the time of learning. **gnuplot** can display graphs in real time.

Gnuplot script example


```
set border lc rgb "white"
set grid lc rgb "white" lt 2
set key opaque box
set object 1 rect behind from screen 0,0 to screen 1,1 fc rgb "#B8B2C3" fillstyle solid
set key right bottom

# smooth [unique, csplines, acsplines, bezier, sbezier]

plot 'accuracy_rate.dat' using 1:2  t "varidation accuracy"  with lines linewidth 2 linecolor rgbcolor "red"
replot 'accuracy_rate.dat' using 1:3  t "test sample accuracy" with lines linewidth 2 linecolor rgbcolor "blue"

pause 10
reread
```

■ Layer setting「LAYER.txt」

LAYER 4 1 [28, 28] *Each layer setting* See layer description END	Number of layers Input feature map from left, input unit width, input unit height ※ Width and height are numbers when input units are regarded as a matrix

■ Describing layers

Fully Connected layer

LAYER_TYPE_FullyConnected/**layerName** **1** [**1**, **10**] **Softmax**	From the left, input of feature map, output unit width, output unit height, start function ※ The width and height are the numbers when looking at the input device as a matrix ※ Activation function described on final page

Convolutional layer

LAYER_TYPE_Convolutional/**layerName** **20** (**5**, **5**) st **1** **ReLU**	From the left, input feature map, convolution width, convolution height, stride, activation function

Convolutional layer

LAYER_TYPE_Convolutional/**layerName** **20** (**5**, **5**) st **1** pd **2** **ReLU**	From the left, input feature map, convolution width, convolution height, stride, padding activation function

DeConvolutional layer

LAYER_TYPE_DeConvolutional/**layerName** **20** (**5**, **5**) st **1** **ReLU**	From the left, input feature map, convolution width, convolution height, stride, activation function

DeConvolutional layer

LAYER_TYPE_DeConvolutional/**layerName** **20** (**5**, **5**) st **1** pd **2** **ReLU**	From the left, input feature map, convolution width, convolution height, stride, padding activation function

maxPooling layer

LAYER_TYPE_maxPooling/**layerName** **20** (**4**, **4**) st **4** **Identity**	From the left, input feature map, convolution width, convolution height, stride, activation function

maxPooling layer

LAYER_TYPE_maxPooling/**layerName** **20** (**4**, **4**) st **4** pd **0** **Identity**	From the left, input feature map, convolution width, convolution height, stride, padding, activation function

AveragePooling layer

LAYER_TYPE_AveragePooling /**layerName** **20** (**4**, **4**) st **4** **Identity**	From the left, input feature map, convolution width, convolution height, stride, activation function

AveragePooling layer

LAYER_TYPE_AveragePooling /**layerName** **20** (**4**, **4**) st **4** pd **0** **Identity**	From the left, input feature map, convolution width, convolution height, stride, padding, activation function

Dropout layer

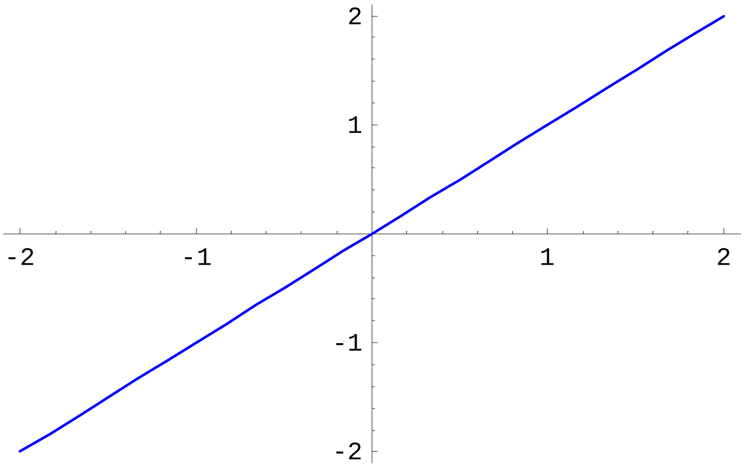
LAYER_TYPE_Dropout/**layerName** **0.5 Identity**	From the left,0.5 is the dropout rate, activation function

BatchNormalize layer

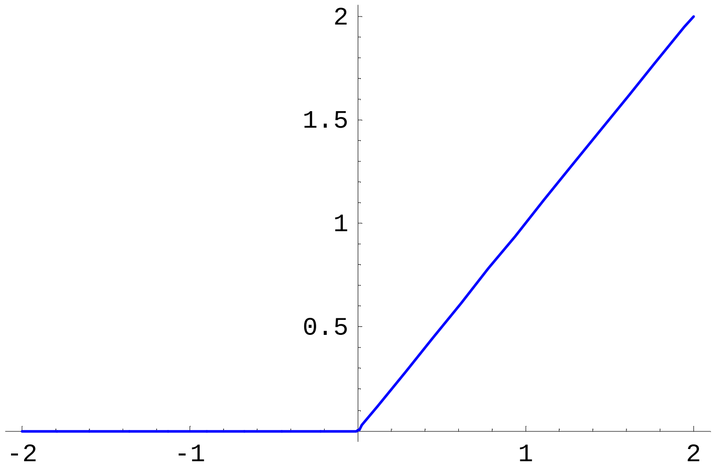
LAYER_TYPE_ BatchNormalize/**layerName** **0.999 Identity**	From the left,0.999 is the dropout decay, activation function

■ **Activation function can use**

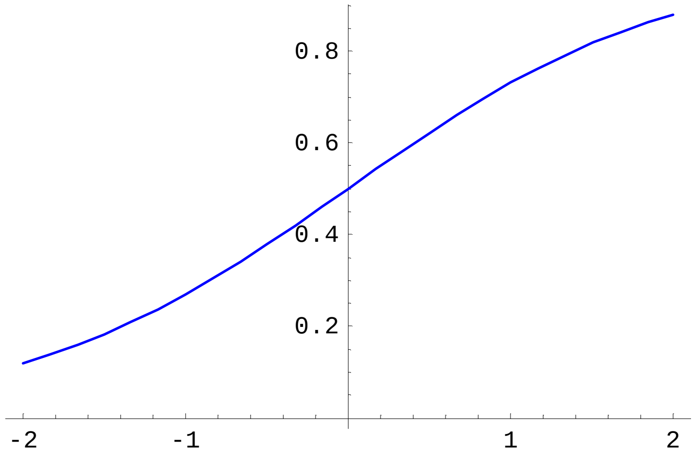
Identity $h(x) = x$



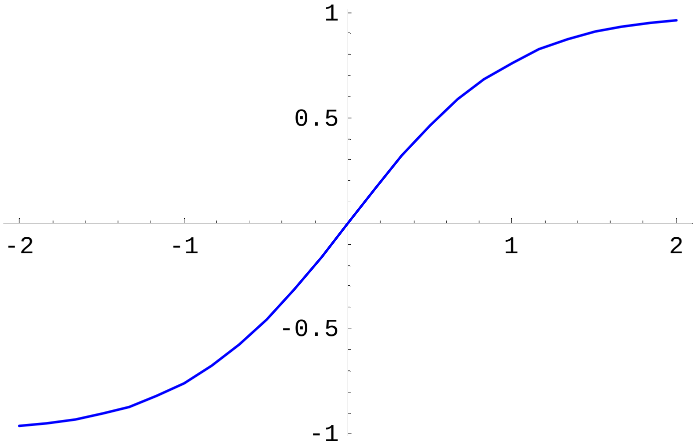
ReLU $h(x) = \max(0, x)$



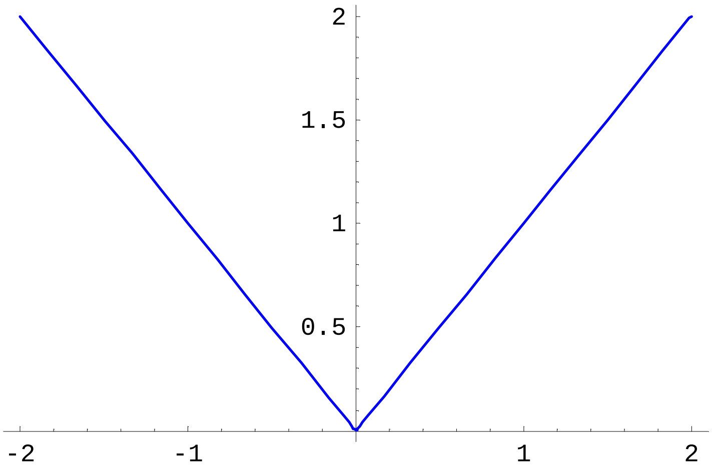
Sigmoid $h(x) = \dfrac{1}{1+e^{-x}} = \dfrac{\tanh(x/2)+1}{2}$



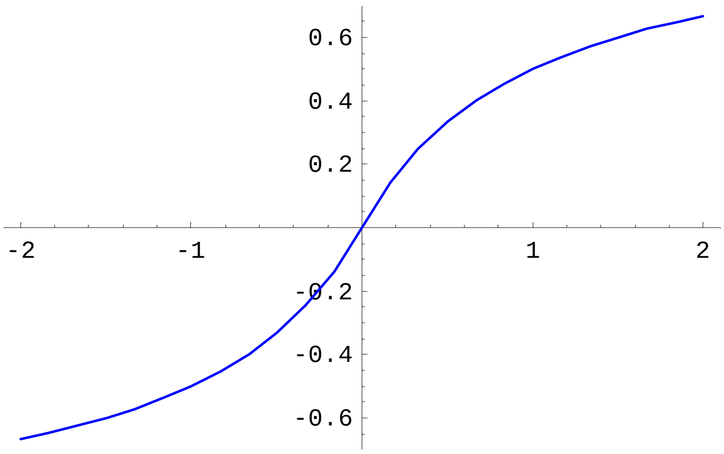
Tanh $h(x) = \tanh(x)$



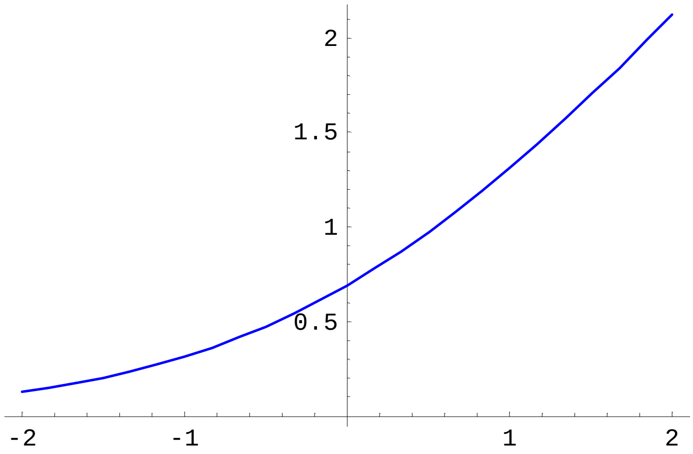
Abs $h(x) = |x|$



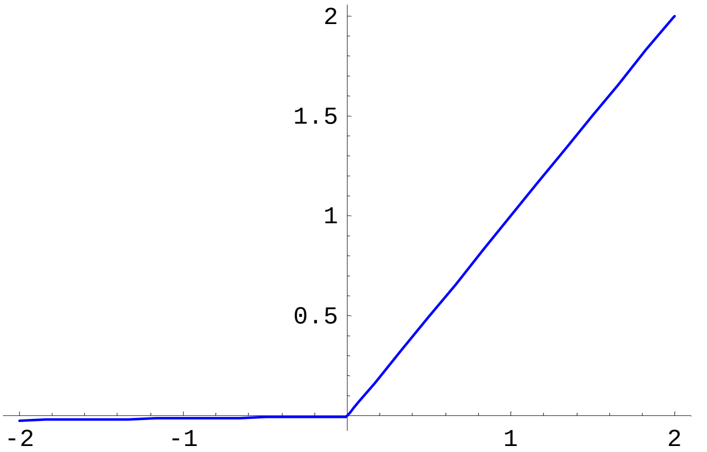
Softsign $h(x) = \dfrac{x}{1+|x|}$



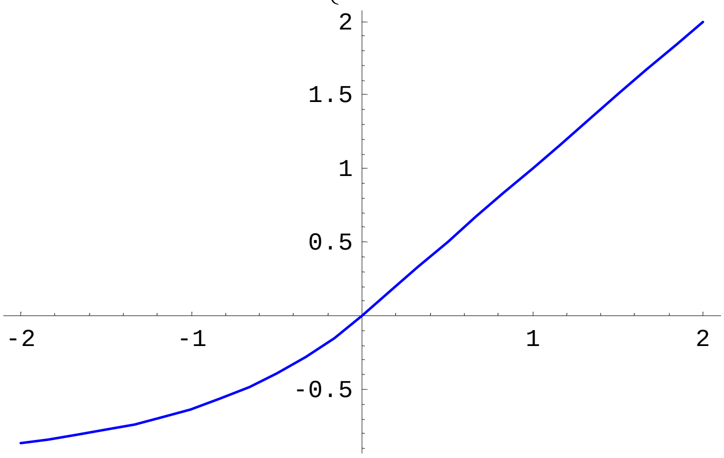
Softplus $$h(x) = \log(1 + e^{x})$$



LReLU $$h(x) = \max(0.01x, x)$$



ELU $$h(x) = \begin{cases} e^{x} - 1 & x < 0 \\ x & x \geq 0 \end{cases}$$



Softmax $$h(x) = \frac{\exp(x)}{\sum_{j=1}^{n} \exp(x_j)}$$

Output UNIT

Activation function	**Loss function**	**Differentiation of loss function**
Identity $h(x) = x$	**Square**	$\frac{\partial E}{\partial w} = y - t$
Softmax $$h(x) = \frac{\exp(x)}{\sum_{j=1}^{n} \exp(x_j)}$$	**CrossEntropy**	$\frac{\partial E}{\partial w} = y - t$

■**Optimizer solver**

Adam

$$m_{t+1} = \beta_1 m_t + (1-\beta_1)\nabla E(\mathbf{w}^t)$$
$$v_{t+1} = \beta_2 v_t + (1-\beta_2)\nabla E(\mathbf{w}^t)^2$$
$$\hat{m} = \frac{m_{t+1}}{1-\beta_1^t}$$
$$\hat{v} = \frac{v_{t+1}}{1-\beta_2^t}$$
$$\mathbf{w}^{t+1} = \mathbf{w}^t - \alpha\frac{\hat{m}}{\sqrt{\hat{v}}+\epsilon}$$

α=0.001,β 1 =0.9,β 2 =0.999,ϵ=10E−8

AdaGrad

$$h_0 = \epsilon$$
$$h_t = h_{t-1} + \nabla E(\mathbf{w}^t)^2$$
$$\eta_t = \frac{\eta_0}{\sqrt{h_t}}$$
$$\mathbf{w}^{t+1} = \mathbf{w}^t - \eta_t \nabla E(\mathbf{w}^t)$$

ϵ=10E−8 ,η 0 =0.001

RMSprop

$$h_t = \alpha h_{t-1} + (1-\alpha)\nabla E(\mathbf{w}^t)^2$$
$$\eta_t = \frac{\eta_0}{\sqrt{h_t}+\epsilon}$$
$$\mathbf{w}^{t+1} = \mathbf{w}^t - \eta_t \nabla E(\mathbf{w}^t)$$

α=0.99,ϵ=10E−8,η0=0.01

AdaDelta

$$h_t = \rho h_{t-1} + (1-\rho)\nabla E(\mathbf{w}^t)^2$$
$$v_t = \frac{\sqrt{s_t+\epsilon}}{\sqrt{h_t+\epsilon}}\nabla E(\mathbf{w}^t)$$
$$s_{t+1} = \rho s_t + (1-\rho)v_t^2$$
$$\mathbf{w}^{t+1} = \mathbf{w}^t - v_t$$

ρ=0.95,ϵ=10E−6

SGD

$$\mathbf{w}^{t+1} \leftarrow \mathbf{w}^t - \eta\frac{\partial E(\mathbf{w}^t)}{\partial \mathbf{w}^t}$$

η=0.01