Q1.1 $\text{softmax}(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}} = \frac{e^{x_i+c}}{\sum_j e^{x_j+c}} = \frac{e^{x_i}}{\sum_j e^{x_j}}$

denote $\vec{y} = \vec{x} + c$, and y_i is ith component of $\vec{y}$

so $\text{softmax}(x+c) = \text{softmax}(x)$

$c = 0$: numerater: $e^{\min(x_i)}$ to $e^{\max(x_i)} \in (1, \infty)$

$c = -\max x_i$ numerater: $e^{\min(x_i)-\max(x_i)}$ $e^0 = 1$

$0 <$ all numerater ≤ 1

$c = -\max x_i$, numerator $e^{x_i - c_i}$ won't blow up. It's more stable.

Q1.2 range: $(0, 1)$

$$\frac{e^{x_i}}{\sum_{j=1} e^{x_j}} = \frac{1}{\sum_{j=1} e^{x_j - x_i}} \qquad 1 < \sum e^{x_j - x_i} < \infty$$

as $e^{x_i - x_i} = 0$

$x_j \in (-\infty, \infty)$ as $j \neq i$

sum: $\frac{\sum e^{x_i}}{\sum e^{x_j}} = 1$

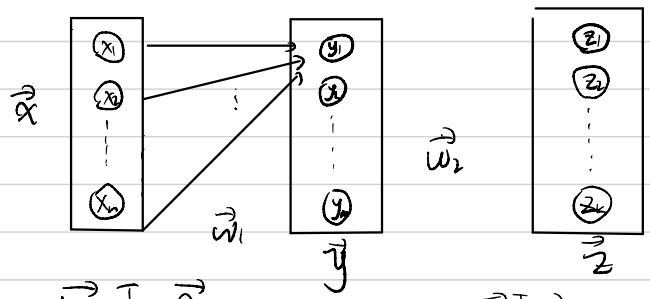
probability mass function

1st step: transfer the data to a positive value

2nd step: calculate the sum for normalization

3rd step: normalize and get the probability

Q1.3



$y_1 = \vec{w}_{11}^T \vec{x}$

$y_m = \vec{w}_{1m}^T \vec{x}$

$w_1 = \begin{pmatrix} \vec{w}_{11}^T \\ \vec{w}_{12}^T \\ \vdots \\ \vec{w}_{1m}^T \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$

$\Rightarrow \vec{y} = w_1 \vec{x}$

when activation function is linear $f(\vec{y}) = f(w_1\vec{x}) = Aw_1\vec{x}$

similarly $\vec{z} = f(w_2\vec{y}) = Aw_2\vec{y} = Aw_2Aw_1\vec{x} = A(w_2Aw_1)\vec{x}$

the equivalent matrix: $B = w_2Aw_1$ $\vec{z} = f(B\vec{x})$

directly from first layer to the third layer.

the same as one layer from $\vec{x}$ to $\vec{z}$.

or multiple layers $A(w_e \cdots Aw_2Aw_1)\vec{x} = Aw\vec{x}$ equivalent to first layer connect to the last layer.

Q1.4 $\sigma(x) = \frac{1}{1+e^{-x}}$

$$\frac{d\sigma(x)}{dx} = \frac{-(-e^{-x})}{(1+e^{-x})^2} = \frac{1+e^{-x}-1}{(1+e^{-x})^2} = \frac{1}{(1+e^{-x})} - \frac{1}{(1+e^{-x})^2} = \sigma(x)(1-\sigma(x))$$

Q1.5

$$\frac{\partial J}{\partial w_{ij}} = \frac{\partial J}{\partial y_j}\frac{\partial y_j}{\partial w_{ij}} = \frac{\partial J}{\partial y_j}x_i$$

$$W^T\vec{x} = \begin{pmatrix} w_{11} & w_{21} & \cdots & w_{d1} \\ w_{12} & w_{22} & \cdots & w_{d2} \\ \vdots & \vdots & & \vdots \\ w_{1k} & & & \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{pmatrix}$$

$$\frac{\partial J}{\partial w} = \begin{pmatrix} \frac{\partial J}{\partial y_1}x_1 & \frac{\partial J}{\partial y_2}x_1 & \cdots & \frac{\partial J}{\partial y_k}x_1 \\ \vdots & \vdots & & 1 \\ \frac{\partial J}{\partial y_1}x_d & \frac{\partial J}{\partial y_2}x_d & \cdots & \frac{\partial J}{\partial y_k}x_d \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{pmatrix}\left(\frac{\partial J}{\partial y_1} \cdots \frac{\partial J}{\partial y_k}\right) = \vec{x}\cdot\delta^T$$

$$\frac{\partial J}{\partial x_i} = \frac{\partial J}{\partial y_1}\frac{\partial y_1}{\partial x_i} + \frac{\partial J}{\partial y_2}\frac{\partial y_2}{\partial x_i} \cdots \frac{\partial J}{\partial y_k}\frac{\partial y_k}{\partial x_i}$$

$$= \frac{\partial J}{\partial y_1}w_{i2} + \frac{\partial J}{\partial y_2}w_{i2} + \cdots \frac{\partial J}{\partial y_k}w_{ik}$$

$$w_i^T\delta \qquad w_i = \begin{pmatrix} w_{i1} \\ w_{i2} \\ \vdots \\ w_{ik} \end{pmatrix}$$

$$\frac{\partial J}{\partial x} = \begin{pmatrix} \frac{\partial J}{\partial x_1} \\ \vdots \\ \frac{\partial J}{\partial x_d} \end{pmatrix} = \begin{pmatrix} w_1^T \\ w_2^T \\ \vdots \\ w_d^T \end{pmatrix} \delta = W\delta$$

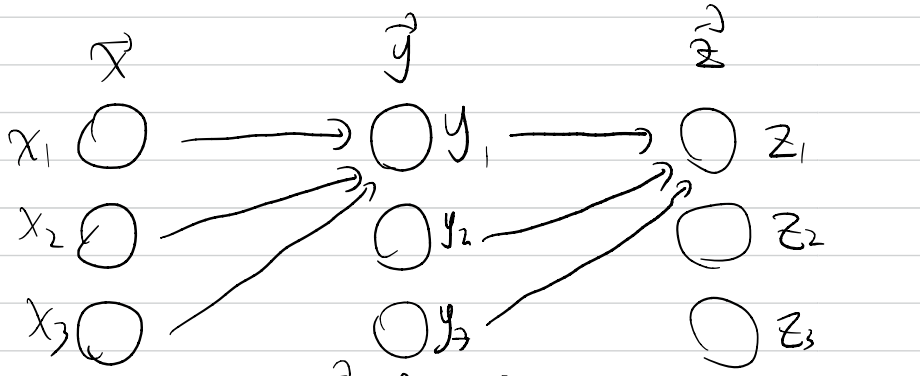
$$\frac{\partial J}{\partial y} = \left(\frac{\partial J}{\partial f(y)}\right) \frac{\partial f(y)}{\partial y}$$

$$\frac{\partial J}{\partial b_j} = \frac{\partial J}{\partial y_j}\frac{\partial y_j}{\partial b_j} = \frac{\partial J}{\partial y_j}$$

$$\frac{\partial J}{\partial b} = \delta$$

Q 1.6 $\frac{d\sigma(x)}{dx} = \sigma(x)(1-\sigma(x))$

1.



$\vec{y} = f(w_1\vec{x})$ $\quad \vec{z} = f(w_2\vec{y})$

denote $\vec{t} = w_1\vec{x}$ $\quad \vec{v} = w_2\vec{y}$

$$\frac{\partial z_1}{\partial y_1} = \frac{\partial z_1}{\partial v_1}\frac{\partial v_1}{\partial y_1} = \sigma(v_1)(1-\sigma(v_1))\frac{\partial v_1}{\partial y_1}$$

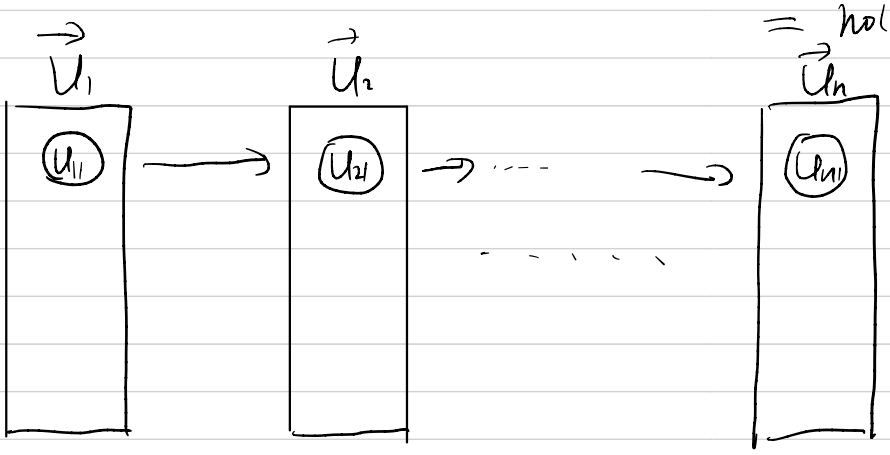
$$\frac{\partial z_1}{\partial x_1} = \frac{\partial z_1}{\partial y_1}\frac{\partial y_1}{\partial x_1} + \frac{\partial z_1}{\partial y_2}\frac{\partial y_2}{\partial x_1} + \frac{\partial z_1}{\partial y_3}\frac{\partial y_3}{\partial x_1}$$

$$\frac{\partial y_1}{\partial x_1} = \frac{\partial y_1}{\partial t_1}\frac{\partial t_1}{\partial x_1} = \sigma(t_1)(1-\sigma(t_1))\frac{\partial t_1}{\partial x_1}$$

So $\frac{\partial z_1}{\partial y_1}\frac{\partial y}{\partial x_1} = \sigma(v_1)(1-\sigma(v_1))\sigma(t_1)(1-\sigma(t_1))\frac{\partial v_1}{\partial y_1}\frac{\partial t_1}{\partial x_1}$

Notice that $\sigma(x)(1-\sigma(x)) \le \left(\frac{\sigma(x)+1-\sigma(x)}{2}\right)^2 = \frac{1}{4}$

$=$ hold when $\sigma(x) = \frac{1}{2}$



$\frac{\partial U_{n1}}{\partial U_{n-1,1}}\frac{\partial U_{n-1,1}}{\partial U_{n-2,1}}\cdots\frac{\partial U_{21}}{\partial U_{11}}$ will contain a coefficient $\le (\frac{1}{4})^{n-1}$

cause the gradient vanish when n is large

2. $\sigma(x) \in (0,1)$

$$\tanh(x) = \frac{1-e^{-2x}}{1+e^{-2x}} = \frac{-(e^{-2x}+1)+2}{1+e^{-2x}} = -1+\frac{2}{1+e^{-2x}}$$

$e^{-2x} \in (0,\infty)$ so $\tanh(x) \in (-1,1)$

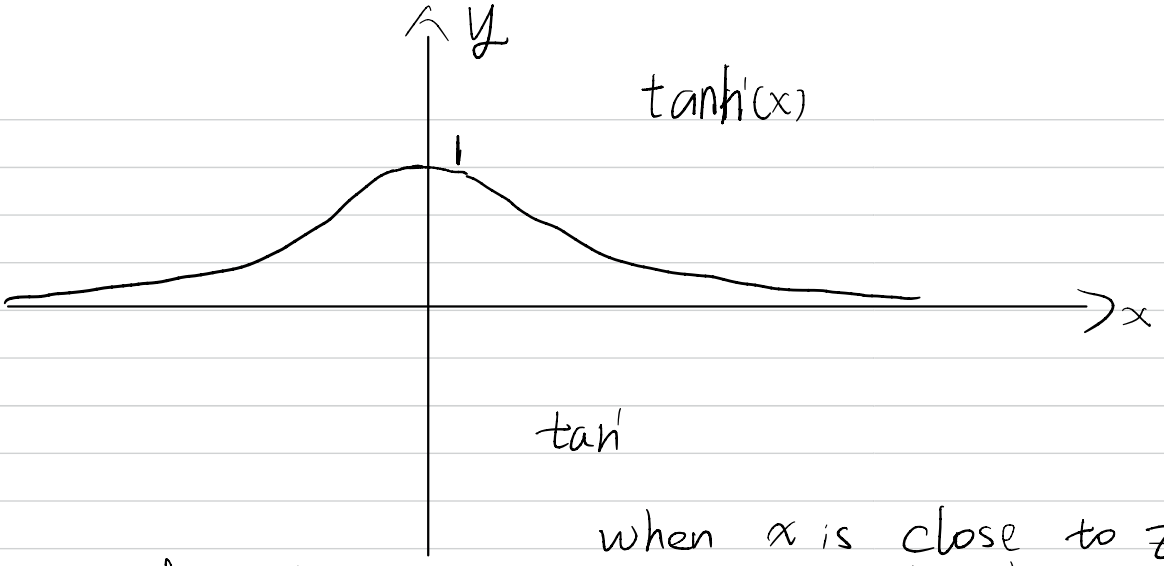
tanh can produce outputs average near zero, while sigmoid produce output that has a mean that is positive.

zero mean makes the normalization for the next layer, and tanh's gradient is big around 0. which makes converge faster

3. $$\frac{d\tanh(x)}{dx} = \frac{-(-2)e^{-2x}(1+e^{-2x})-(1-e^{-2x})(-2e^{-2x})}{(1+e^{-2x})^2} = \frac{4e^{-2x}}{(1+e^{-2x})^2}$$

$$\tanh(x) = \frac{1-e^{-2x}}{1+e^{-2x}} = \frac{(1+e^{-2x})^2-(1-e^{-2x})^2}{(1+e^{-2x})^2}$$

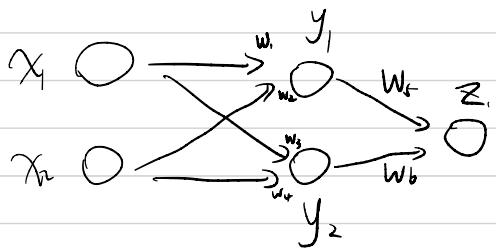
$$\tanh^2(x) = \frac{(1-e^{-2x})^2}{(e^{-2x}+1)^2} = 1-\tanh^2(x)$$



when x is close to zero, the gradient is near 1, it won't make the gradient $\times \frac{1}{4}$ this makes the model converge faster, and it doesn't go to zero. $G(x) = 0$ when $x = 0$ or 1.

4. $G(x) = \frac{\tanh(\frac{1}{2}x)+1}{2}$

Q.2.11



omit the activation function gradient.

after input: $y_1 = y_2 = z_1 = 0$

$\frac{\partial z_1}{\partial y_1} = w_5 = 0$ $\frac{\partial z_1}{\partial y_2} = w_6 = 0$

$\frac{\partial z_1}{\partial w_5} = y_1 = 0$ $\frac{\partial z_1}{\partial w_6} = y_2 = 0$

$\frac{\partial y_1}{\partial x_2} = w_2 = 0$

$\frac{\partial y_1}{\partial x_1} = w_1 = 0$

$\frac{\partial y_2}{\partial x_1} = w_3 = 0$

$\frac{\partial y_2}{\partial x_2} = w_4 = 0$

$\frac{\partial y_1}{\partial w_1} = x_1$

$\frac{\partial y_1}{\partial w_2} = x_2$

$\frac{\partial y_2}{\partial w_3} = x_1$

$\frac{\partial y_2}{\partial w_4} = x_2$

only parameter in the first layer is trained. the training will be very slow. after that $w_1 = w_3$, $w_2 = w_4$

so $y_1 = y_2$ still hold in the afterward trainings

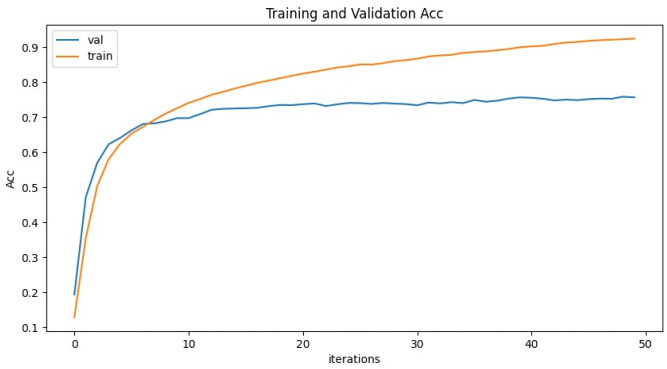
$$\frac{\partial z}{\partial w_5} = y_1 = y_2 = \frac{\partial z}{\partial w_6}$$

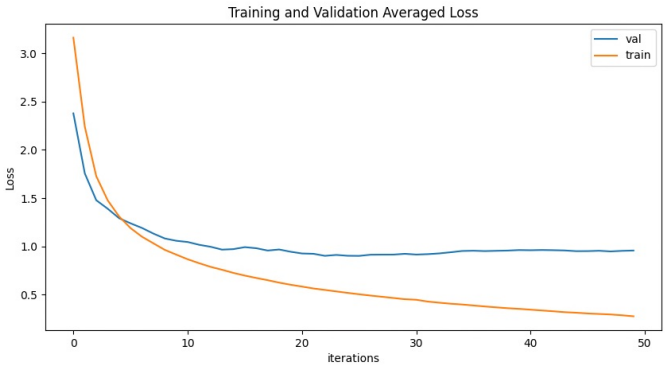
the network will be symmetric and the outputs for different features will be almost the same.

Q2.1.3 If all the inputs are the same, it will be similar to Q2.1.1 the network will become symmetric. Random initialization prevents the weights to be the same.

With proposed initialization, activation value be more stable between each layer. Back-propagated gradients become smaller without scaling depending on layer size. The gradients of very different magnitudes at different layers may yield to ill-conditioning and slower training.
The proposed method's variances of gradient in each layer are almost the same. This prevent gradient vanishing.

Q3.1.1



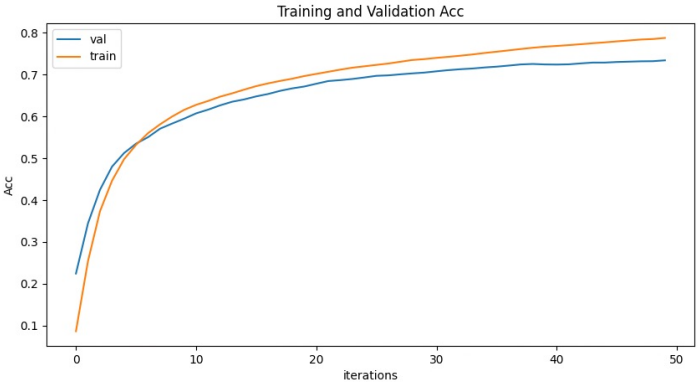


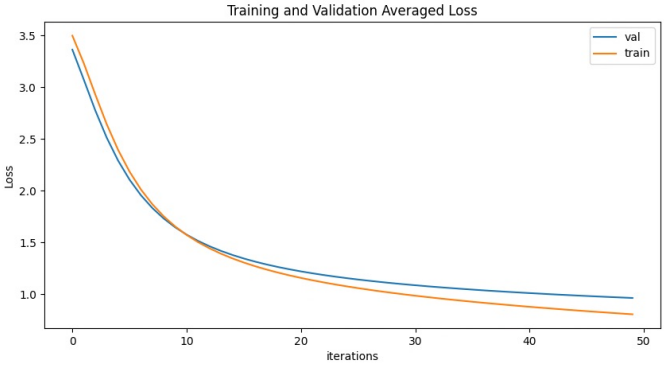
Acc and loss with learning rate = 0.01

best lr = 0.07 with validation acc = 0.757

Q3.1.2

Acc and loss with learning rate = 0.1





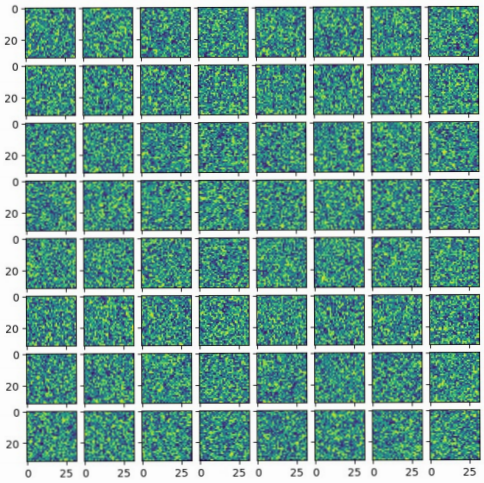
Acc and loss with learning rate = 0.001

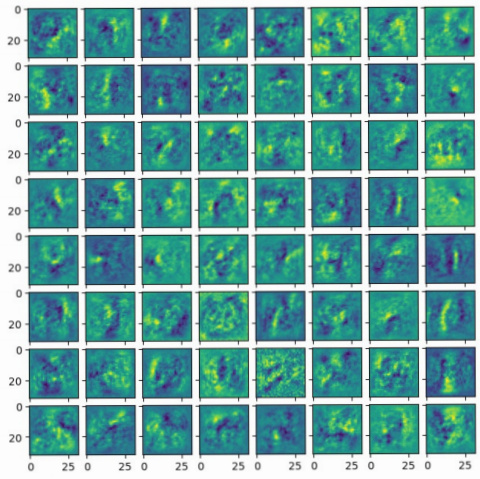
The best acc is in Q3.1.1

lr too big, network doesn't converge, the changes of weights are too big to reach an optimal point.

lr too small, converge very slowly. You need more time to converge

Q3.1.3





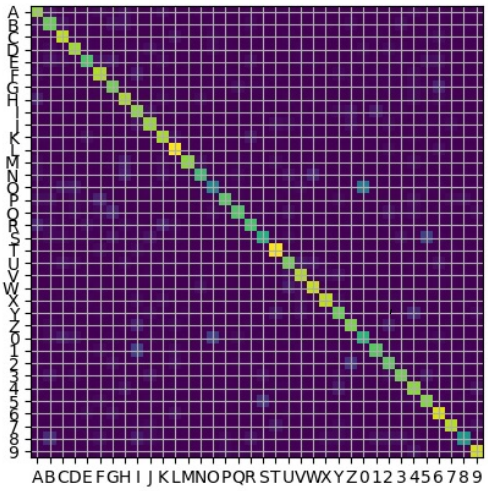
top: Visualization of weights before training

down: Visualization after training

right after initialization the weights are just noise.

after training, some regular shapes appear.

Q3.1.4

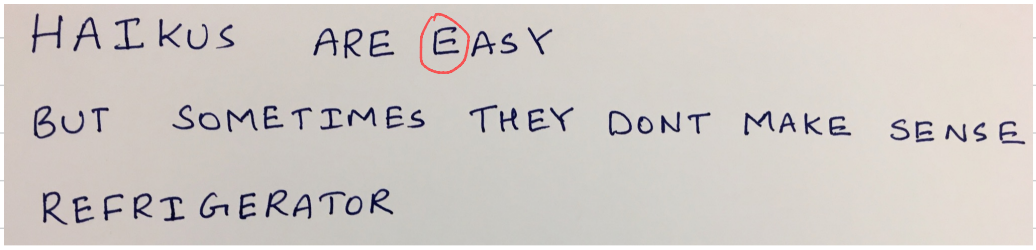


number 0 and character O. They look very similar

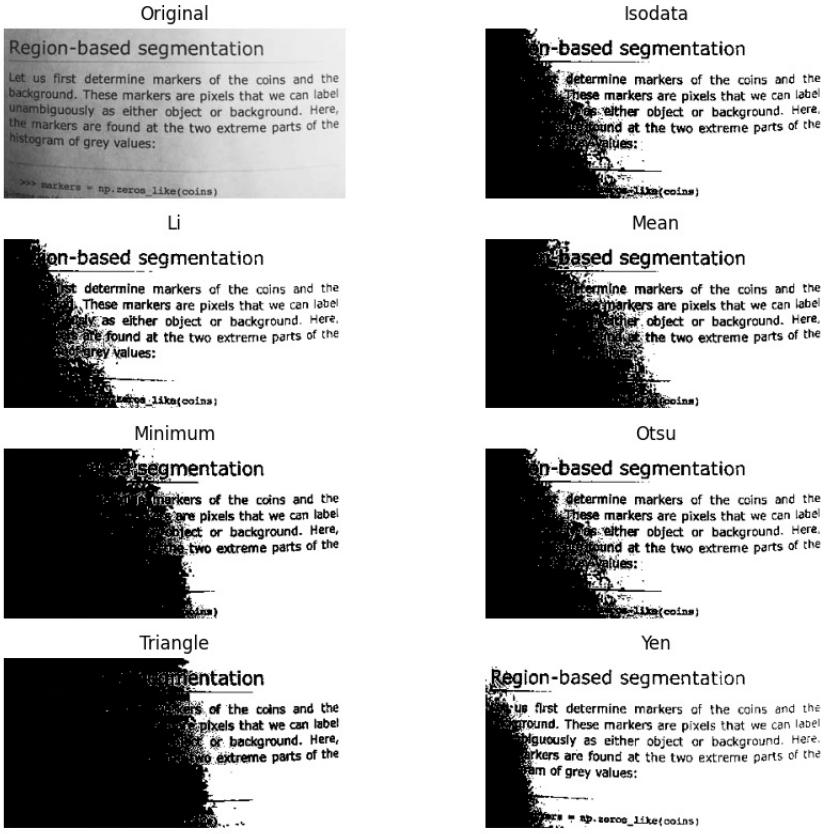
5 - s I - 1 those pairs look very similar

Q4.1 1. The foreground only contain text. Nothing else will be recognized as foreground.

2. The hand writting is pretty good. Each character or number exactly form one connected component.

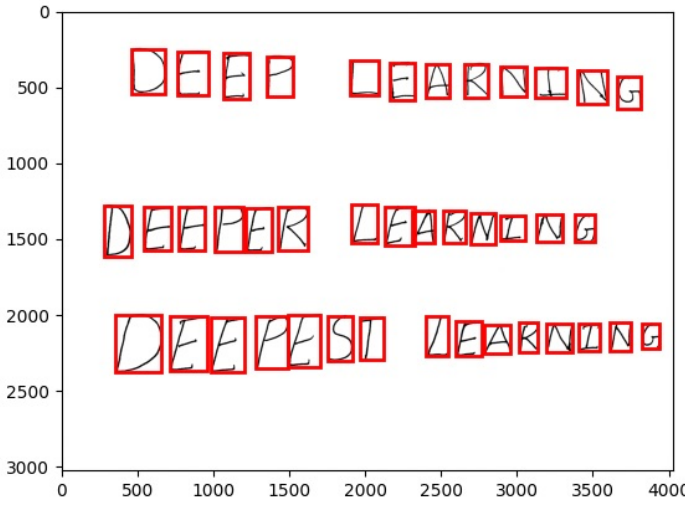


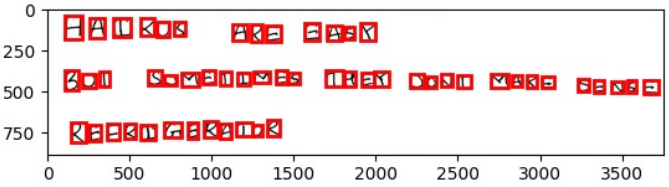
letter "E" is not one connected component. "E" and "-" will be the two component

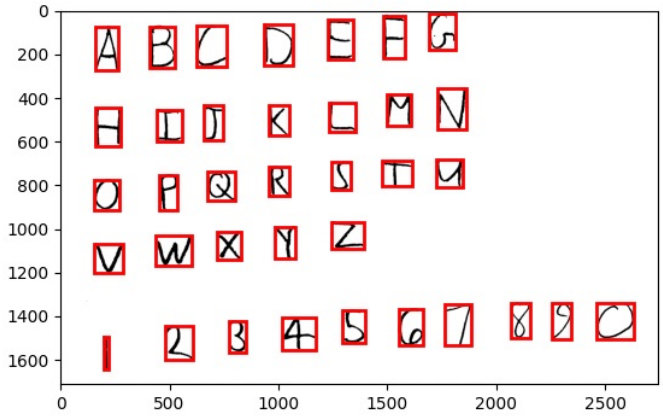


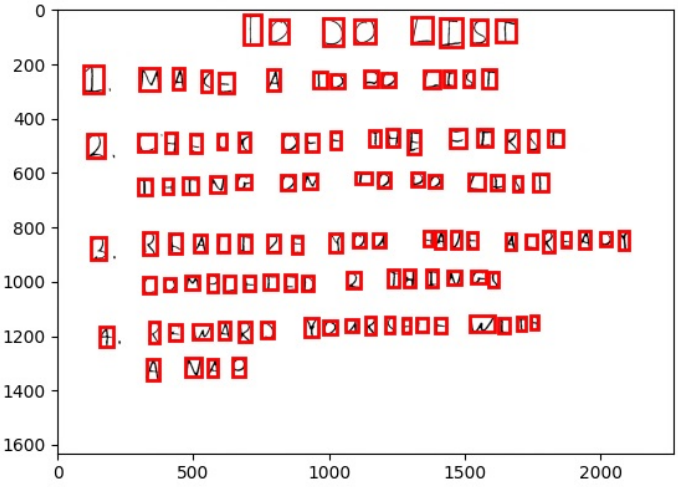
The bending of paper makes some part of the paper recognized as all in foreground. (Left part) This make bounding box responses to non-characters.

Q4.3









Q4.4

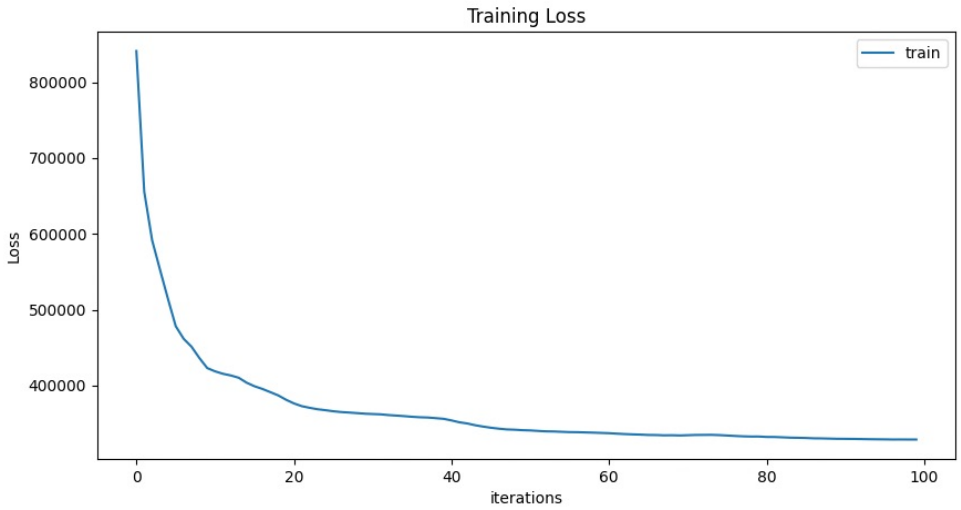
```
T0DQLI5T
IHAKEATDDQLI5T
2CHKCK0FFTHEFIRST
THING0NT0DQLIST
3REALIZEY0UHAUEALRGADT
C0MPLETLDZTHINGS
4REWARDY0URSELFWITH
ANAP
0.8173913043478261
```

```
HAIKU5AREGAGY
GUTSQMETIMESTREYD0NTMAKGBGNGE
RBFRIGERATOR
0.7777777777777778
```

```
ABCDEFG
HIJKLMN
0PQRSTU
VWXYZ
IZ3GSG789Q
0.8055555555555556
```

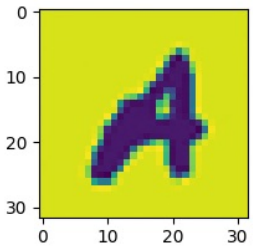
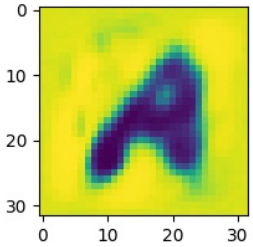
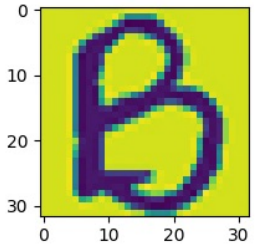
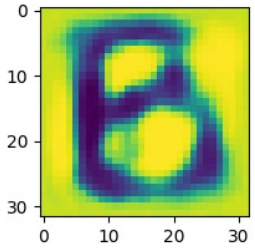
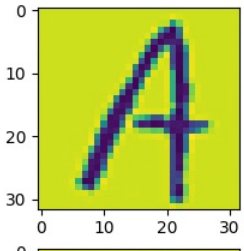
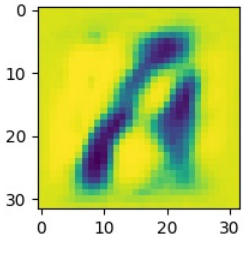
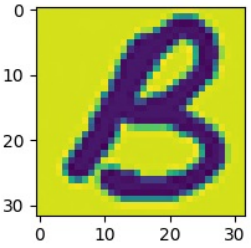
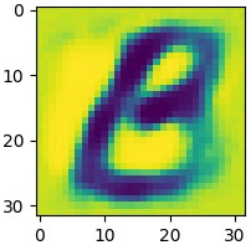
```
DEEPLEARMING
DEBPERLEARNING
DEEPESTLEARNIHG
0.926829268292683
```

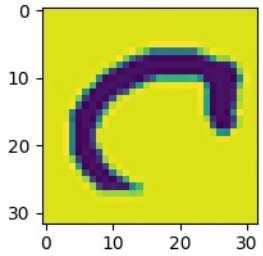
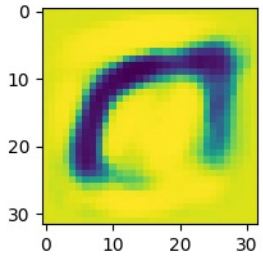
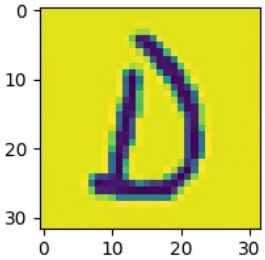
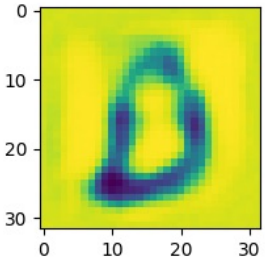
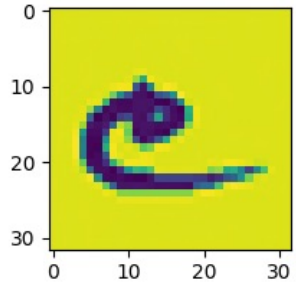
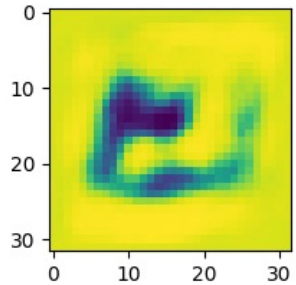
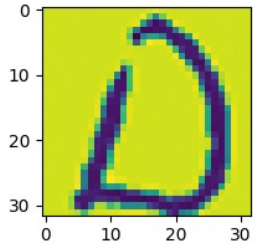
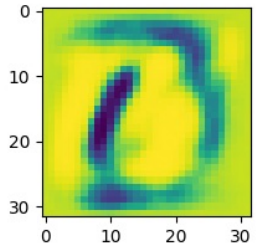
Q5.2



The loss drop quickly in the first 20 iterations and decay slowly in the rest iterations.

Q5.3.1

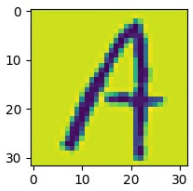
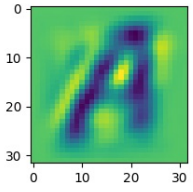
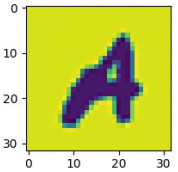
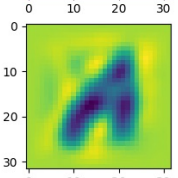









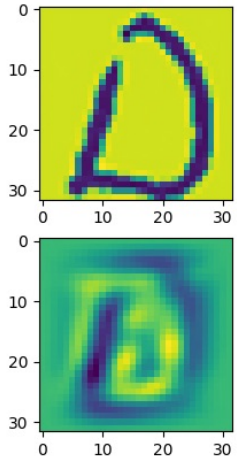
In reconstruction images, some details are lost, the image is blurred, and the posisition of the pixel is moved.

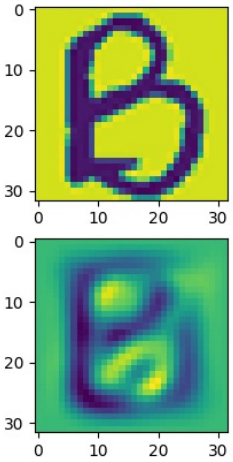
Q5.2.2 Avg PSNR = 15.3568

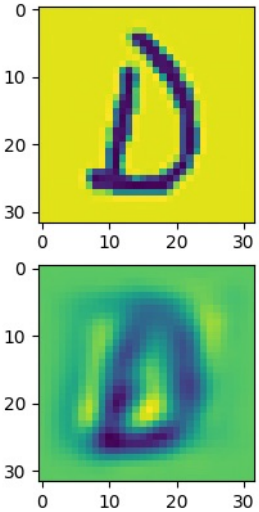
Q6.1 32 x 1024 rank = 32

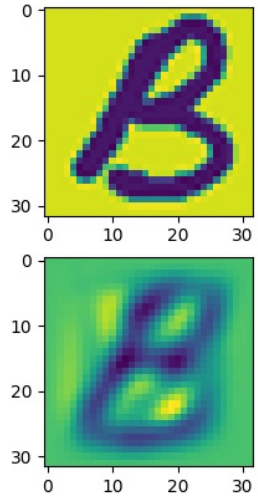
Q6.2

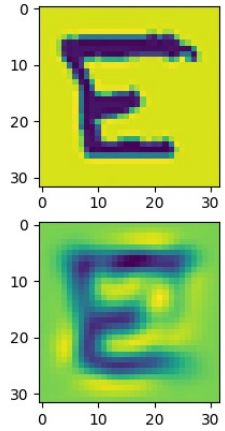





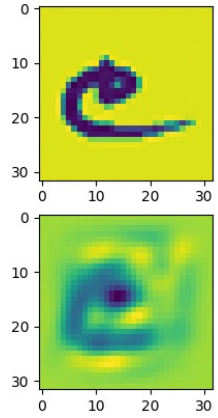


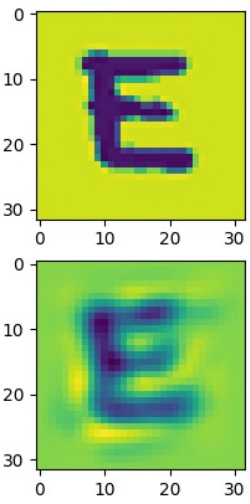


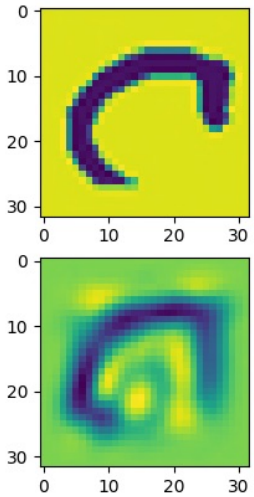












The reconstructed image is very blurry compared to original ones.

The one using autoencoder looks better than PCA

Q 6.3 psnr = 16.284

It is better than auto encoder.

because position affects psnr very much. pixels in autoencoder-reconstructed images might moved from original places so psnr is higher doesn't mean image looks better.

Q 6.4 $1024 \times 32 + 32$

$+32 \times 32 + 32 = 68704$

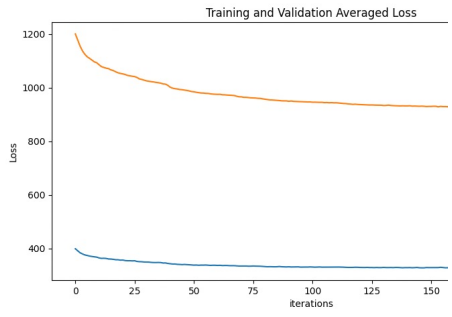
auto $+32 \times 32 + 32$

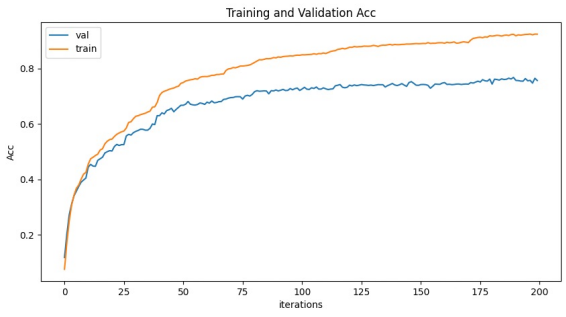
encoder: $+32 \times 1024 + 1024$

PCA: $1024 \times 32 = 32768$

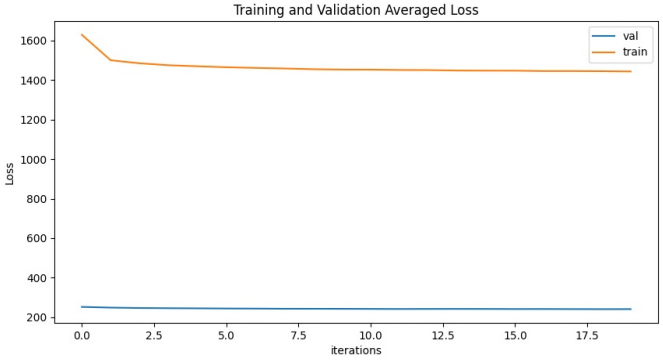
The auto encoder learns more, with linear and non-linear transforms, so it looks better.

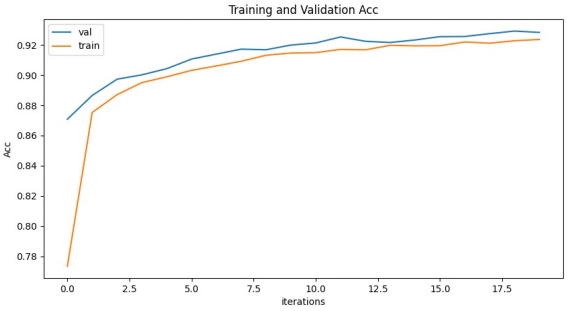
Q 7.1.1



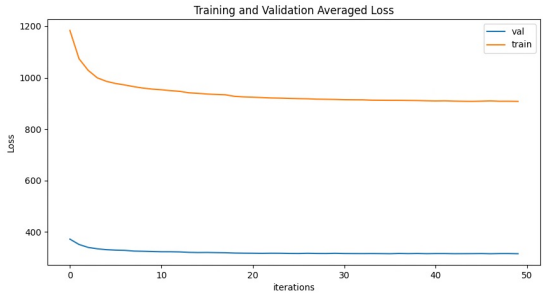


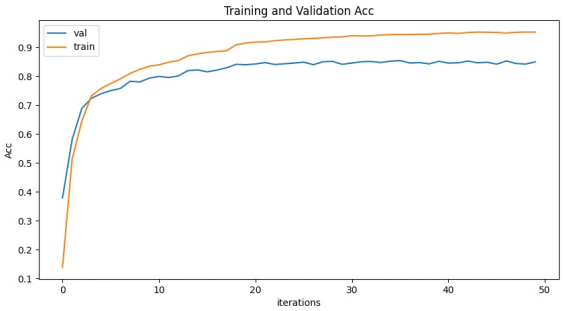
Q7.1.2





Q7.1.3





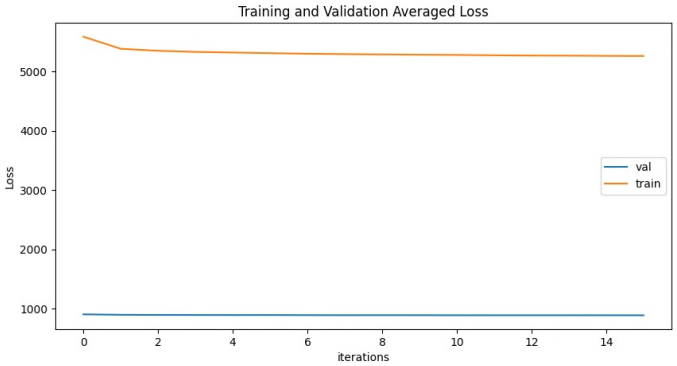
Q7.1.4

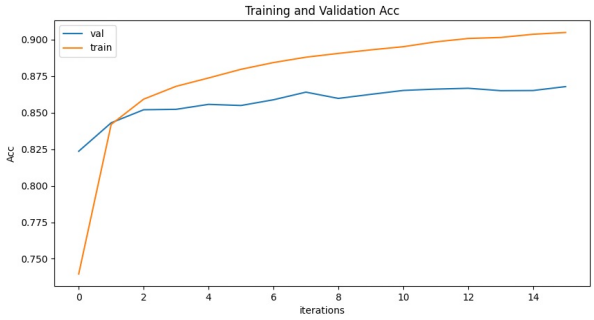
```
ABCDEFG
HIJKLMN
0PQRSTU
VWXYZ
IZ3GSG789Q
0.8055555555555556
```

```
T0DQLI5T
IHAKEATDDQLI5T
2CHKCK0FFTHEFIRST
THING0NT0DQLIST
3REALIZEY0UHAUEALRGADT
C0MPLETLDZTHINGS
4REWARDY0URSELFWITH
ANAP
0.8173913043478261
```

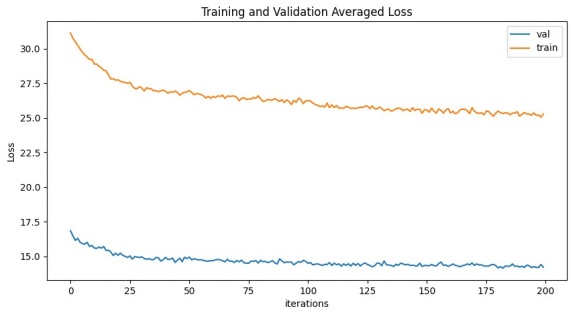
```
DEEPLEARMING
DEBPERLEARNING
DEEPESTLEARNIHG
0.926829268292683
```

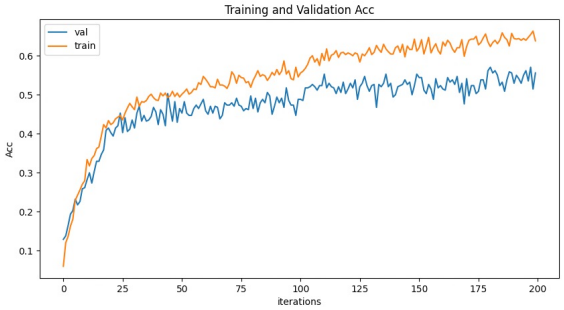
```
HAIKU5AREGAGY
GUTSQMETIMESTREYD0NTMAKGBGNGE
RBFRIGERATOR
0.7777777777777778
```





Q7.2.1 use flower 17.





train from scratch acc and loss

fine tuning squeezenet1_1 acc and loss

self designed model converges much more slower and acc is lower