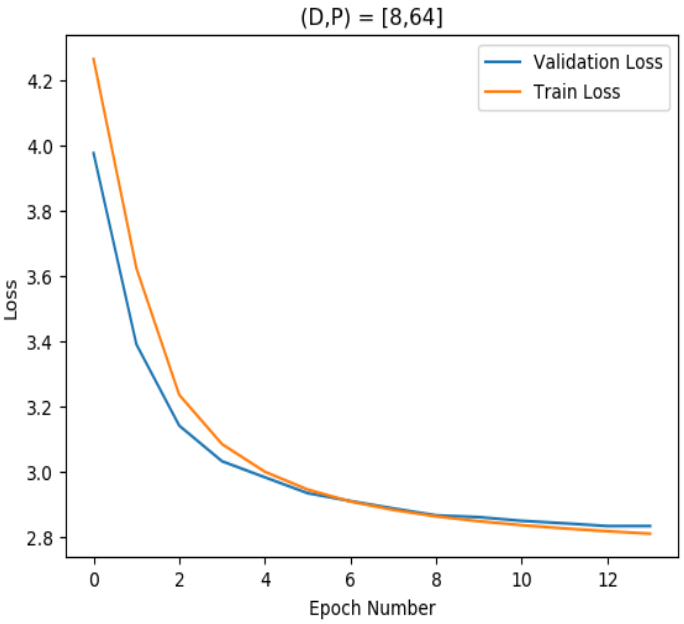
Some Important Notes :

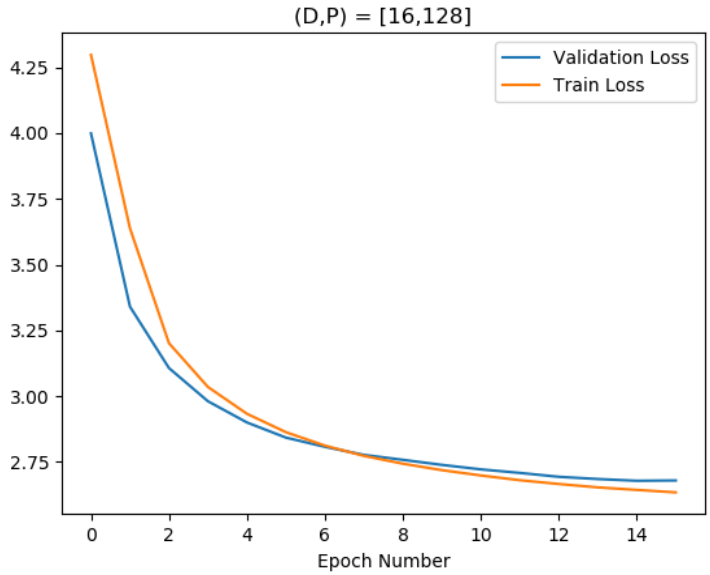
- I implement first question in python that the tester could run code '**python sait_akturk_21501734_hw3.py 1**' therefore the tester should get this libraries h5py, matplotlib, sklearn, numpy
  Install anaconda first then 'conda install matplotlib' , 'conda install h5py' , 'conda install scikit-learn' , 'conda install numpy'.

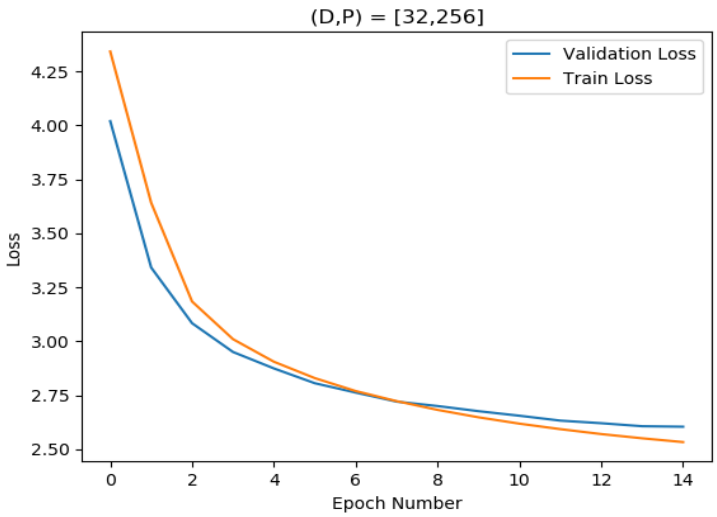
- I implement second question in matlab therefore the tester should run code in matlab (in python file, you should copy and paste to .m file)environment '**sait_akturk_21501734_hw3.m 2'**

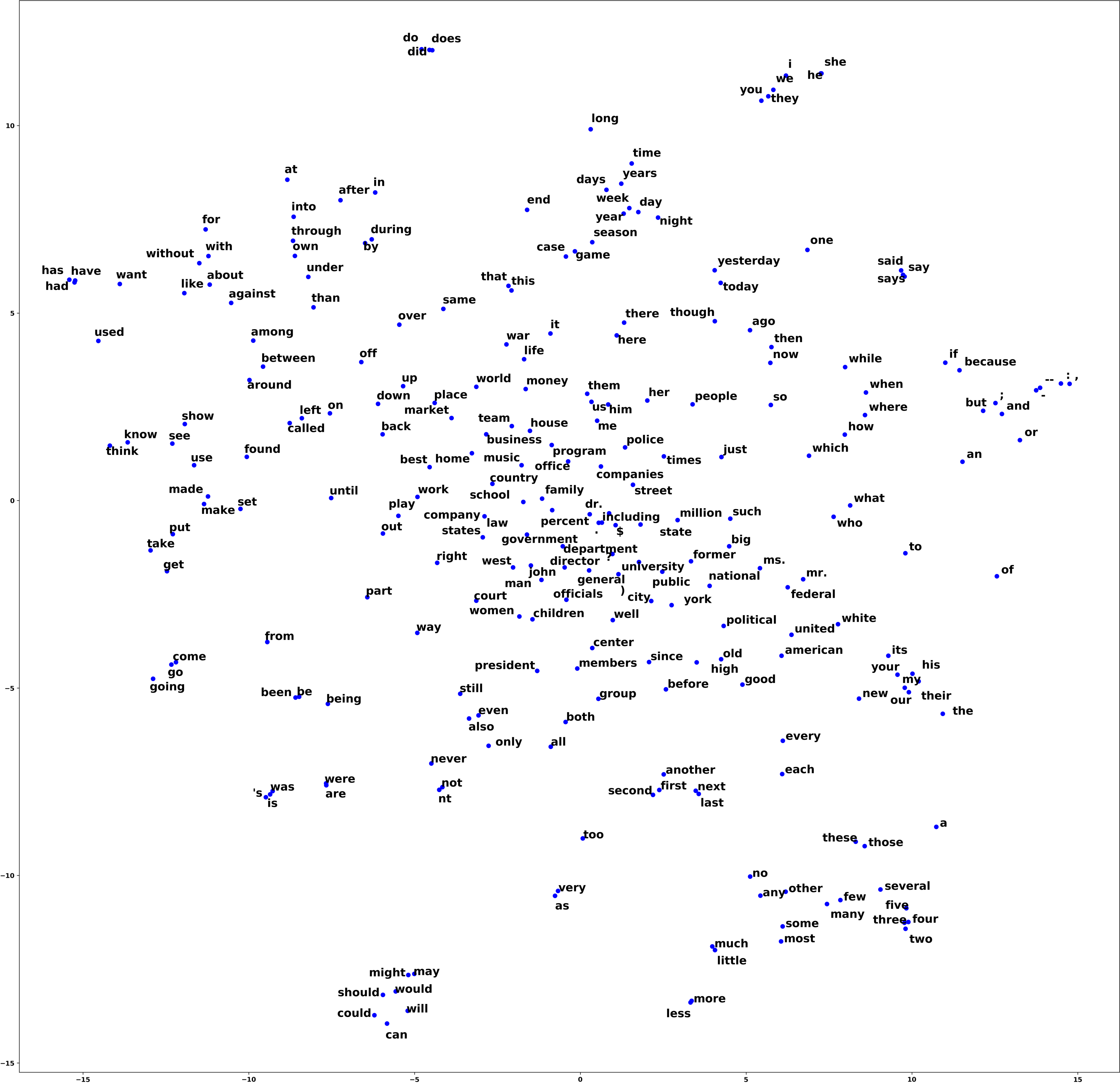
QUESTION 1

A) For bigger size of hidden unit help to reduce cross entropy loss and increase accuracy and creates tendency for over-fitting faster, TSNE results become more reasonable results.









do
does
did
i
she
we
he
you
they
long
time
years
at
after
in
days
week
day
end
into
year
night
for
through
during
season
with
own
by
case
one
without
game
yesterday
has
have
want
under
about
said
say
had
like
that
this
says
today
against
than
same
there
though
ago
over
used
among
it
war
here
then
off
between
life
now
while
if
because
--
:
,
up
world
money
them
around
her
people
so
when
down
place
us
him
-
but
;
and
left
on
market
where
show
team
house
called
me
back
how
know
see
business
police
or
program
just
think
found
which
an
use
best
home
music
times
office
companies
country
made
until
work
family
school
street
set
make
what
dr.
play
million
such
company
including
percent
law
who
out
put
states
$
state
.
government
take
big
to
department
right
?
former
director
west
university
get
ms.
john
of
general
mr.
public
national
man
)
court
part
officials
city
york
federal
women
children
well
political
white
way
united
from
center
its
come
since
old
american
president
members
his
your
go
high
going
my
good
before
still
been
be
group
new
their
being
our
the
even
both
also
only
all
every
never
another
each
were
not
first
next
second
was
's
are
nt
is
last
too
a
these
those
no
very
any
other
several
as
few
five
many
three
four
some
two
most
much
little
might
may
would
should
will
could
more
can
less
10
5
0
−5
−10
−15
−15
−10
−5
0
5
10
15

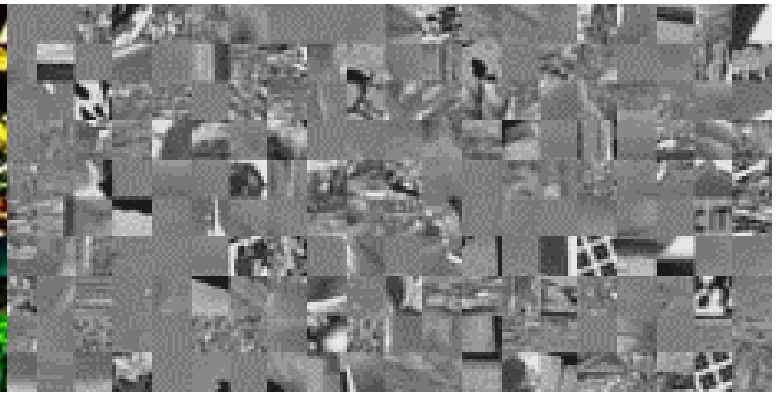
B) Since some words are easy to determine similarity like ‘he she it’ or ‘can could may’ we can see that these words that used in similar context are in same clusters and we can identify similar words or words that have similar meanings. Our embedding model find good result, however some specific words like ‘john’ , ‘president’ used in sentences have no similar words or word that similar meanings. However, embedding does not looking only similarity also find frequency of words used in some trigram structure.

C) I randomly select 5 triagram from test data and get words that 10 maximum probability that I found in test prediction. (1. column is the most likely fourth word, probability drop, when we go to 10. column). I think some triagrams are sensible in the context of fourth word. And I notice that the similar words that we found part b, generally, becomes each other as a prediction because since embedding is also finding similarity between words, the fourth words are generally close the each other in the context of meaning and the place that they are used in sentences.

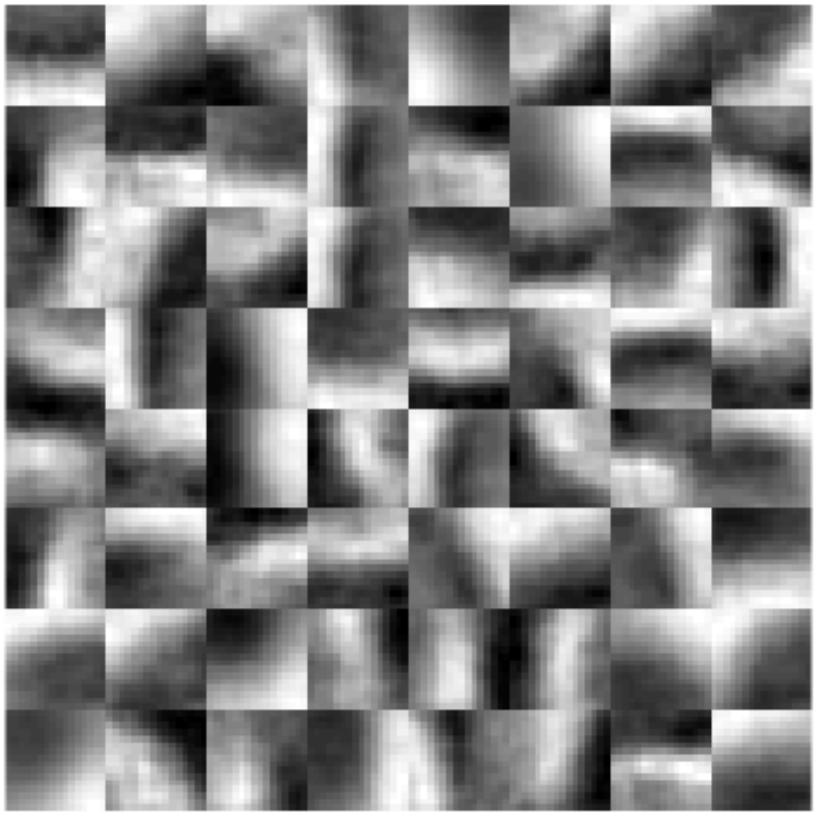
0	1	2	3	4	5	6	7	8	9	10
think you are	going	.	not	here	in	right	there	good	on	out
and play my	game	own	day	case	time	life	way	last	best	home
work and not	.	the	,	for	last	like	at	a	right	have
that 's been	the	going	a	there	good	here	like	all	with	right
did not like	it	that	this	them	to	the	him	you	me	.

QUESTION 2

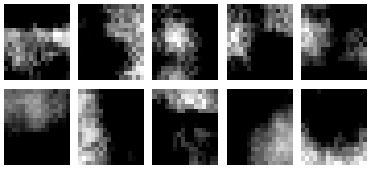
A) I randomly select 200 different patches from the data and when I did the normalization I lost color feature therefore as you can see , we could only detect different textures like edge horizontal or diagonal etc. Therefore, I should expect to see in learned features in part b should give this basic features. Since color information is lost the patches that only one color and textures is completely lost due to normalization.



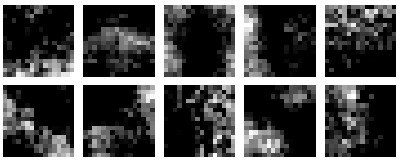
B-C) Our purpose find good features that could represent normalized patches textures, therefore, I experiment with different rho and beta values for code and I found that rho=0.01 and and beta = 2.5 as optimal results. In the beginning I try to reduce final loss that we got from fmincgrad. However, the features that I found became to overfit with data just represent some patches. Therefore, I believe I should find features that could detect basic orientation and illuminance difference like Gabor and first derivate filters. Therefore, I made tradeoff between loss and quality of features. In the end, I got this features that almost represents all oriented gradients inan image(for edge detection for every orientation).



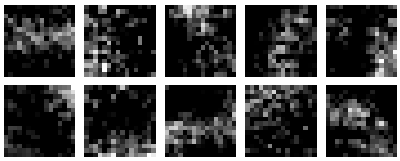
Iteration = 50, lambda = 0,hidden =10



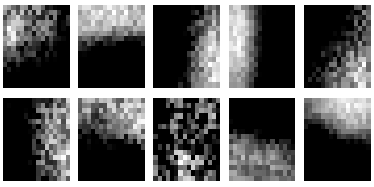
Iteration = 225, lambda = 0,hidden =10



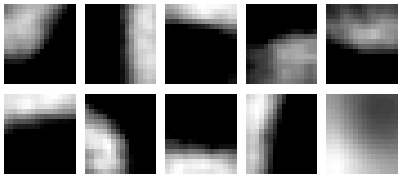
Iteration = 400, lambda = 0,hidden =10



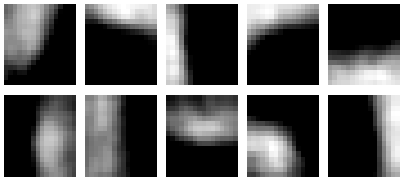
Iteration = 50, lambda = 1e-4,hidden =10



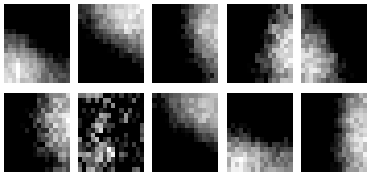
Iteration = 225, lambda = 1e-4,hidden =10



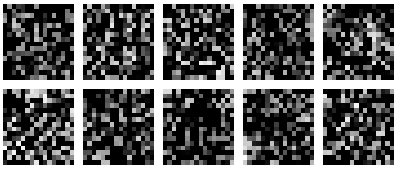
Iteration = 400, lambda = 1e-4,hidden =10



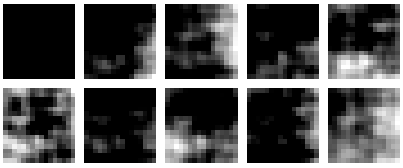
Iteration = 50, lambda = 0.001,hidden =10



Iteration = 225, lambda = 0.001,hidden =10

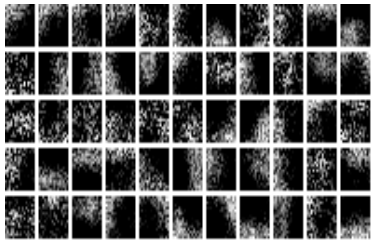


Iteration = 400, lambda = 0.001,hidden =10

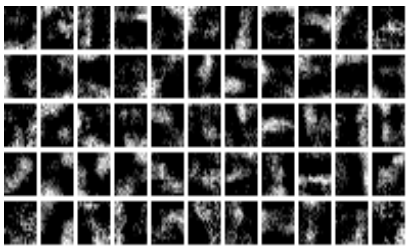


Without Regularization the features from patches have not smooth transition due to overfitting the data. And also we can observe that when we we increase iteration number the features get more and more sharp transitions. And One another thing is that hidden layer number is small and not representative the features for image patches because as we can see the learned features have not all basic oriented gradient that an optimal features should have.

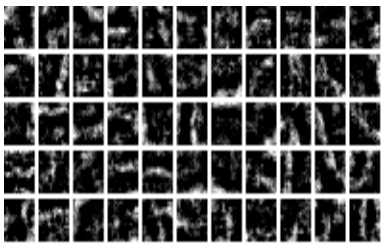
Iteration = 50, lambda = 0,hidden =55



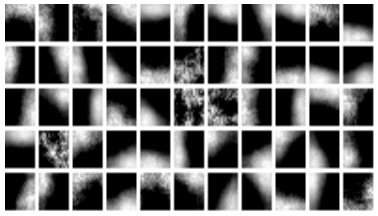
Iteration = 225, lambda = 0,hidden =55



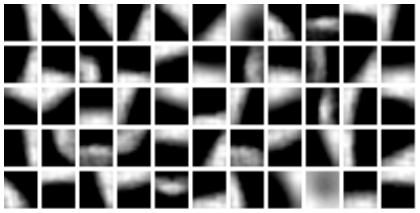
Iteration = 400, lambda = 0,hidden =55



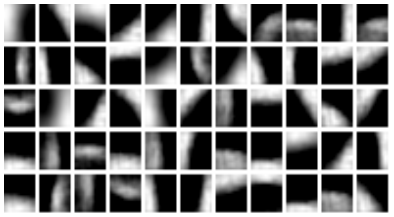
Iteration = 50, lambda = 1e-4,hidden =55



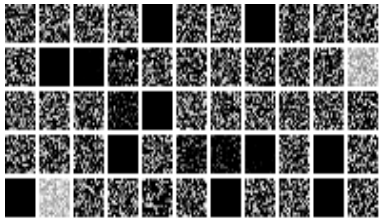
Iteration = 225, lambda = 1e-4,hidden =55



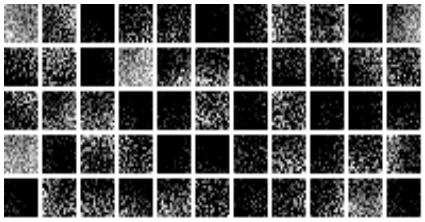
Iteration = 400, lambda = 1e-4,hidden =55



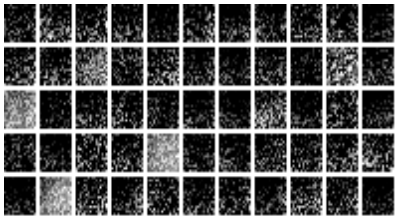
Iteration = 50, lambda = 1e-3,hidden =55



Iteration = 225, lambda = 1e-3,hidden =55

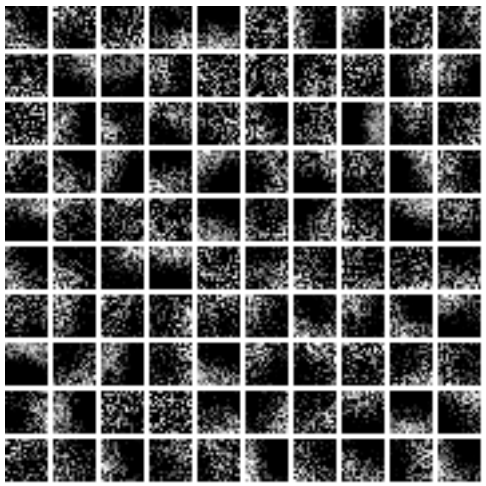


Iteration = 400, lambda = 1e-3,hidden =55

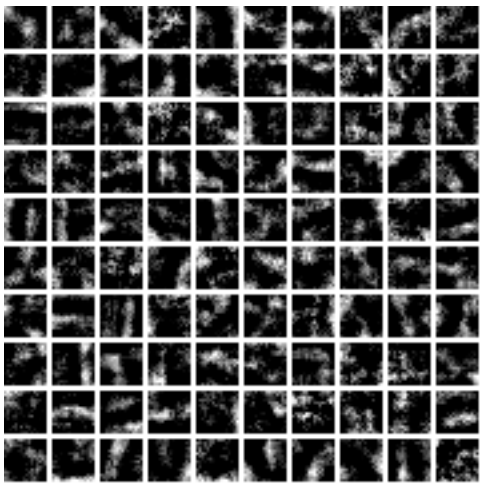


55 hidden units are more representative of basic oriented gradients in image than 10 hidden units. And we can make same observations that the more iterations creates more sharp transition due to memorization of features. And more l2 regularization create more smothed transition of gradients and over regularization creates noise like due to penalty of regularization of learning.

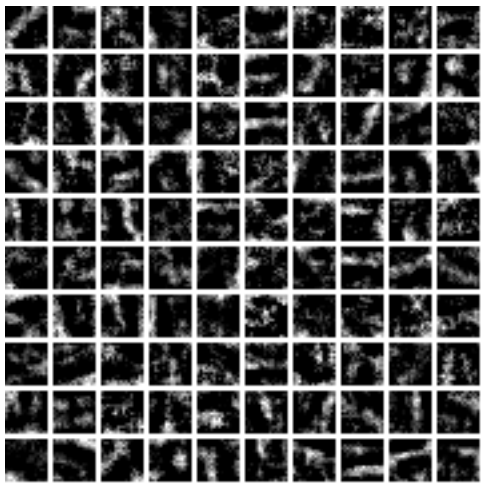
Iteration = 50, lambda = 0,hidden = 100



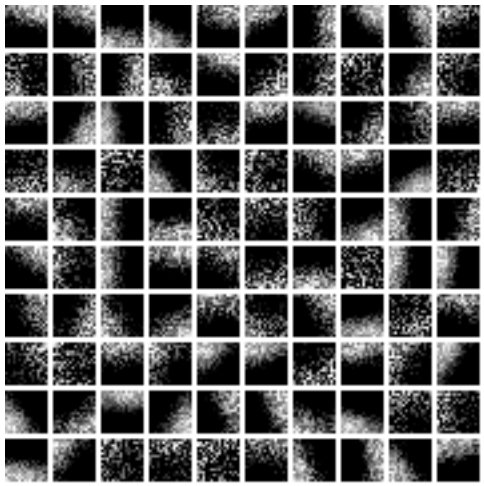
Iteration = 225, lambda = 0,hidden = 100



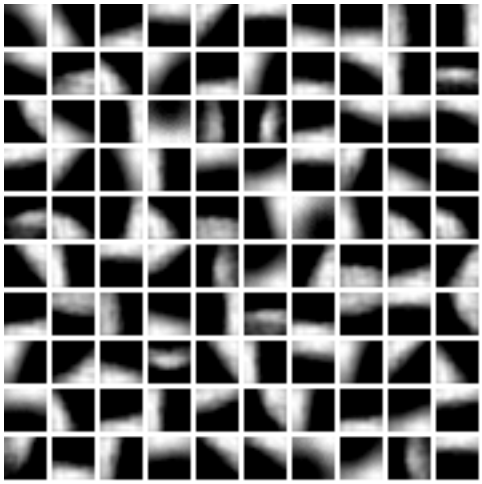
Iteration = 400, lambda = 0,hidden = 100



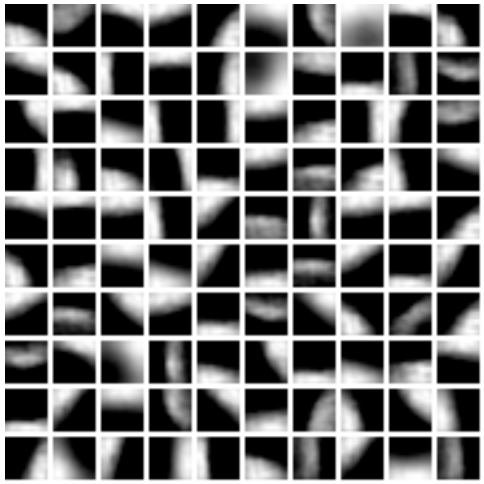
Iteration = 50, lambda = 1e-4,hidden = 100



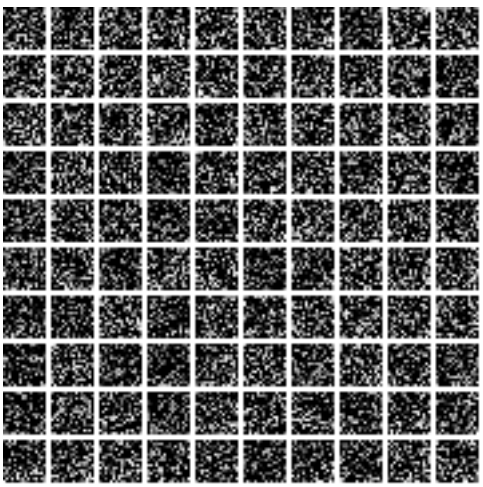
Iteration = 225, lambda = 1e-4,hidden = 100



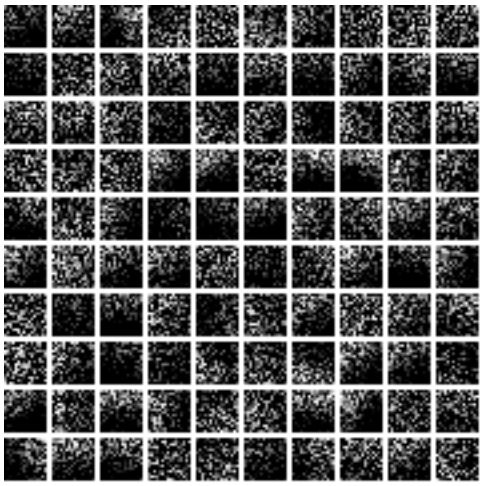
Iteration = 400, lambda = 1e-4,hidden = 100



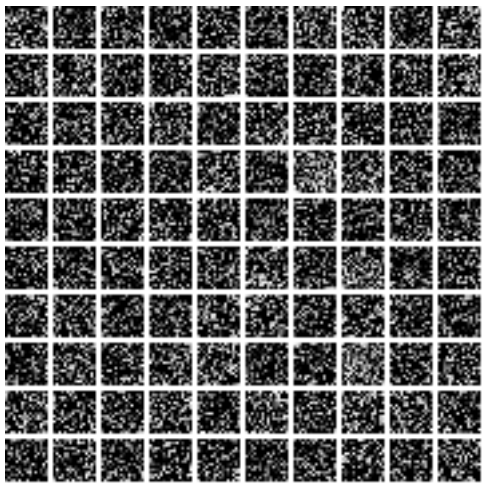
Iteration = 50, lambda = 1e-3,hidden = 100



Iteration = 225, lambda = 1e-3,hidden = 100



Iteration = 400, lambda = 1e-3,hidden = 100



100 hidden units are over representative of patches that some features learned again and again. Same observation could observed here that more iteration makes features transition of gradients sharp and overfit, and more l2 regularization cause to prevent learn anything for bigger lambda not in middle images.