Assignment1

November 10, 2021

1 Assignment 1. Music Century Classification

Assignment Responsible: Natalie Lang.

In this assignment, we will build models to predict which **century** a piece of music was released. We will be using the "YearPredictionMSD Data Set" based on the Million Song Dataset. The data is available to download from the UCI Machine Learning Repository. Here are some links about the data:

- https://archive.ics.uci.edu/ml/datasets/yearpredictionmsd
- http://millionsongdataset.com/pages/tasks-demos/#yearrecognition

Note that you are note allowed to import additional packages **(especially not PyTorch)**. One of the objectives is to understand how the training procedure actually operates, before working with PyTorch's autograd engine which does it all for us.

1.1 Question 1. Data (21%)

Start by setting up a Google Colab notebook in which to do your work. Since you are working with a partner, you might find this link helpful:

- https://colab.research.google.com/github/googlecolab/colabtools/blob/master/notebooks/colab-github-demo.ipynb

The recommended way to work together is pair coding, where you and your partner are sitting together and writing code together.

To process and read the data, we use the popular `pandas` package for data analysis.

[ ]:
```
import pandas
import numpy as np
import matplotlib.pyplot as plt
np.random.seed(1) # To reproduce our experiment
```

Now that your notebook is set up, we can load the data into the notebook. The code below provides two ways of loading the data: directly from the internet, or through mounting Google Drive. The first method is easier but slower, and the second method is a bit involved at first, but can save you time later on. You will need to mount Google Drive for later assignments, so we recommend figuring how to do that now.

Here are some resources to help you get started:

- http.://colab.research.google.com/notebooks/io.ipynb

[ ]:
```
load_from_drive = False

if not load_from_drive:
  csv_path = "http://archive.ics.uci.edu/ml/machine-learning-databases/00203/
↪YearPredictionMSD.txt.zip"
else:
  from google.colab import drive
  drive.mount('/content/gdrive')
  csv_path = '/content/gdrive/My Drive/YearPredictionMSD.txt.zip' # TODO -␣
↪UPDATE ME WITH THE TRUE PATH!

t_label = ["year"]
x_labels = ["var%d" % i for i in range(1, 91)]
df = pandas.read_csv(csv_path, names=t_label + x_labels)
```

Now that the data is loaded to your Colab notebook, you should be able to display the Pandas DataFrame `df` as a table:

[ ]:
```
df
```

[ ]:
```
        year     var1      var2  ...     var88      var89     var90
0       2001  49.94357  21.47114  ... -1.82223  -27.46348   2.26327
1       2001  48.73215  18.42930  ... 12.04941   58.43453  26.92061
2       2001  50.95714  31.85602  ... -0.05859   39.67068  -0.66345
3       2001  48.24750  -1.89837  ...  9.90558  199.62971  18.85382
4       2001  50.97020  42.20998  ...  7.88713   55.66926  28.74903
...      ...       ...       ...  ...       ...        ...       ...
515340  2006  51.28467  45.88068  ...  3.42901  -41.14721 -15.46052
515341  2006  49.87870  37.93125  ... 12.96552   92.11633  10.88815
515342  2006  45.12852  12.65758  ... -6.07171   53.96319  -8.09364
515343  2006  44.16614  32.38368  ... 20.32240   14.83107  39.74909
515344  2005  51.85726  59.11655  ... -5.51512   32.35602  12.17352

[515345 rows x 91 columns]
```

To set up our data for classification, we'll use the "year" field to represent whether a song was released in the 20-th century. In our case `df["year"]` will be 1 if the year was released after 2000, and 0 otherwise.

[ ]:
```
df["year"] = df["year"].map(lambda x: int(x > 2000))
```

[ ]:
```
df.head(20)
```

[ ]:
```
    year      var1      var2  ...     var88      var89     var90
0      1  49.94357  21.47114  ...  -1.82223  -27.46348   2.26327
1      1  48.73215  18.42930  ...  12.04941   58.43453  26.92061
2      1  50.95714  31.85602  ...  -0.05859   39.67068  -0.66345
3      1  48.24750  -1.89837  ...   9.90558  199.62971  18.85382
4      1  50.97020  42.20998  ...   7.88713   55.66926  28.74903
5      1  50.54767   0.31568  ...   5.00283  -11.02257   0.02263
6      1  50.57546  33.17843  ...   4.50056   -4.62739   1.40192
```

```
7       1  48.26892    8.97526  ...  -0.30633     3.98364  -3.72556
8       1  49.75468   33.99581  ...   5.48708    -9.13495   6.08680
9       1  45.17809   46.34234  ...  11.49326   -89.21804 -15.09719
10      1  39.13076  -23.01763  ...  -2.59543   109.19723  23.36143
11      1  37.66498  -34.05910  ...  -5.19009     8.83617  -7.16056
12      1  26.51957 -148.15762  ...  23.00230  -164.02536  51.54138
13      1  37.68491  -26.84185  ...  14.11648 -1030.99180  99.28967
14      0  39.11695   -8.29767  ...  -2.14942  -211.48202 -12.81569
15      1  35.05129  -67.97714  ...  20.73063  -562.07671  43.44696
16      1  33.63129  -96.14912  ...   3.44539   259.10825  10.28525
17      0  41.38639  -20.78665  ...   4.44627    58.16913  -0.02409
18      0  37.45034   11.42615  ...  25.73235   157.22967  38.70617
19      0  39.71092   -4.92800  ...   2.34002   -31.57015   1.58400

[20 rows x 91 columns]
```

1.1.1 Part (a) -- 7%

The data set description text asks us to respect the below train/test split to avoid the "producer effect". That is, we want to make sure that no song from a single artist ends up in both the training and test set.

Explain why it would be problematic to have some songs from an artist in the training set, and other songs from the same artist in the test set. (Hint: Remember that we want our test accuracy to predict how well the model will perform in practice on a song it hasn't learned about.)

[ ]:
```
df_train = df[:463715]
df_test = df[463715:]

# convert to numpy
train_xs = df_train[x_labels].to_numpy()
train_ts = df_train[t_label].to_numpy()
test_xs = df_test[x_labels].to_numpy()
test_ts = df_test[t_label].to_numpy()

# Write your explanation here
'''
To evaluate our model properly, we want to know how well the model will perform␣
 ↪over a new songs that it hasn't encountered before.
Using songs from the same artist for both the training set and the test set can␣
 ↪lead to a situation in which the model learns to extract features based on␣
 ↪the singer,
e.g., the singer's voice, and will decide based on those features.
In this scenario, the model might perform well on samples of songs in the test␣
 ↪where their corresponding singers also appeared in the train set.
But this doesn't guarantee that the model will generalize for songs in the test␣
 ↪set that their singers didn't appear in the train set.
'''
```

[ ]:
```
"\nTo evaluate our model properly, we want to know how well the model will
perform over a new songs that it hasn't encountered before.\nUsing songs from
the same artist for both the training set and the test set can lead to a
situation in which the model learns to extract features based on the
singer,\ne.g., the singer's voice, and will decide based on those features.\nIn
this scenario, the model might perform well on samples of songs in the test
where their corresponding singers also appeared in the train set.\nBut this
doesn't guarantee that the model will generalize for songs in the test set that
their singers didn't appear in the train set.\n"
```

1.1.2 Part (b) -- 7%

It can be beneficial to **normalize** the columns, so that each column (feature) has the *same* mean and standard deviation.

[ ]:
```
feature_means = df_train.mean()[1:].to_numpy() # the [1:] removes the mean of␣
 ↪the "year" field
feature_stds  = df_train.std()[1:].to_numpy()

train_norm_xs = (train_xs - feature_means) / feature_stds
test_norm_xs = (test_xs - feature_means) / feature_stds
```

Notice how in our code, we normalized the test set using the *training data means and standard deviations*. This is *not* a bug.

Explain why it would be improper to compute and use test set means and standard deviations. (Hint: Remember what we want to use the test accuracy to measure.)

[ ]:
```
# Write your explanation here
"""
We normalize the data with respect to the training set to make the input data␣
 ↪used for training "more identical distributed" during training.
We are using test data to evaluate our model performance over new unseen data.
If we are using the average and standard deviation of this test set to␣
 ↪normalize it, we enforce dependency between different samples in the test␣
 ↪set.
This harms the idea of model evaluation over new unseen data, e.g., if we␣
 ↪evaluate our model for two different test sets, both contain some identical␣
 ↪samples.
Following this approach will lead to different outcomes for the same input␣
 ↪sample (because the average and the standard deviation will be different),␣
 ↪which demonstrates the problem with this approach.
"""
```

[ ]:
```
'\nWe normalize the data with respect to the training set to make the input data
used for training "more identical distributed" during training.\nWe are using
test data to evaluate our model performance over new unseen data.\nIf we are
using the average and standard deviation of this test set to normalize it, we
enforce dependency between different samples in the test set. \nThis harms the
idea of model evaluation over new unseen data, e.g., if we evaluate our model
```

```
for two different test sets, both contain some identical samples.\nFollowing
this approach will lead to different outcomes for the same input sample (because
the average and the standard deviation will be different), which demonstrates
the problem with this approach.\n'
```

1.1.3 Part (c) -- 7%

Finally, we'll move some of the data in our training set into a validation set.

Explain why we should limit how many times we use the test set, and that we should use the validation set during the model building process.

[ ]:
```
# shuffle the training set
reindex = np.random.permutation(len(train_xs))
train_xs = train_xs[reindex]
train_norm_xs = train_norm_xs[reindex]
train_ts = train_ts[reindex]

# use the first 50000 elements of `train_xs` as the validation set
train_xs, val_xs           = train_xs[50000:], train_xs[:50000]
train_norm_xs, val_norm_xs = train_norm_xs[50000:], train_norm_xs[:50000]
train_ts, val_ts           = train_ts[50000:], train_ts[:50000]

# Write your explanation here
'''
If we will not limit the times that we are using the test set, we may encounter␣
 ↪overfitting phenomena over the test data,
which in turn will lead to good preformance over the test set, this phenomena␣
 ↪may occur becuase we are tuning our model to achieve
good preformance over a specific set of test samples.
This is a problem because the main reason that we are using a test set is to␣
 ↪evaluate our model preformance over an unseen data,
i.e., we test how well the training procedure will be able to generalize for␣
 ↪unseen samples.
We can avoid this issue by examing our model preformance (during the bulding␣
 ↪procedure) via
another set which we call "validation" set. The validation dataset is different␣
 ↪from the test dataset,
both datasets are held back from the training of the model,but the validation␣
 ↪set used to give an unbiased preformance estimation
of the final tuned model when comparing between different final models."
'''
```

[ ]:
```
'\nIf we will not limit the times that we are using the test set, we may
encounter overfitting phenomena over the test data, \nwhich in turn will lead to
good preformance over the test set, this phenomena may occur becuase we are
tuning our model to achieve\ngood preformance over a specific set of test
samples. \nThis is a problem because the main reason that we are using a test
```

```
set is to evaluate our model preformance over an unseen data,\ni.e., we test how
well the training procedure will be able to generalize for unseen samples. \nWe
can avoid this issue by examing our model preformance (during the bulding
procedure) via\nanother set which we call "validation" set. The validation
dataset is different from the test dataset, \nboth datasets are held back from
the training of the model,but the validation set used to give an unbiased
preformance estimation\nof the final tuned model when comparing between
different final models."\n'
```

1.2 Part 2. Classification (79%)

We will first build a *classification* model to perform decade classification. These helper functions are written for you. All other code that you write in this section should be vectorized whenever possible (i.e., avoid unnecessary loops).

[ ]:
```python
def sigmoid(z):
  return 1 / (1 + np.exp(-z))

def cross_entropy(t, y):
  epsilon=1e-15 #to avoid log(0)
  return -t * np.log(y+epsilon) - (1 - t) * np.log(1 - y+epsilon)

def cost(y, t):
  return np.mean(cross_entropy(t, y))

def get_accuracy(y, t):
  acc = 0
  N = 0
  for i in range(len(y)):
    N += 1
    if (y[i] >= 0.5 and t[i] == 1) or (y[i] < 0.5 and t[i] == 0):
      acc += 1
  return acc / N
```

1.2.1 Part (a) -- 7%

Write a function `pred` that computes the prediction `y` based on logistic regression, i.e., a single layer with weights `w` and bias `b`. The output is given by:

$$y = \sigma(\mathbf{w}^T\mathbf{x} + b), \tag{1}$$

where the value of y is an estimate of the probability that the song is released in the current century, namely year $= 1$.

[ ]:
```python
def pred(w, b, X):
  """
  Returns the prediction `y` of the target based on the weights `w` and scalar
  bias `b`.
```

```
    Preconditions: np.shape(w) == (90,)
                   type(b) == float
                   np.shape(X) = (N, 90) for some N

    >>> pred(np.zeros(90), 1, np.ones([2, 90]))
    array([0.73105858, 0.73105858]) # It's okay if your output differs in the␣
  ↪last
    decimals
    """
    return sigmoid(np.dot(X,w)+b)
```

1.2.2 Part (b) -- 7%

Write a function `derivative_cost` that computes and returns the gradients $\frac{\partial \mathcal{L}}{\partial \mathbf{w}}$ and $\frac{\partial \mathcal{L}}{\partial b}$. Here, `X` is the input, `y` is the prediction, and `t` is the true label.

```
[ ]: def derivative_cost(X, y, t):
    """
    Returns a tuple containing the gradients dLdw and dLdb.

    Precondition: np.shape(X) == (N, 90) for some N
                  np.shape(y) == (N,)
                  np.shape(t) == (N,)

    Postcondition: np.shape(dLdw) = (90,)
             type(dLdb) = float
    """
    N = np.shape(y)[0]
    dLdb = np.mean(y - t)
    dLdw = 1/N * np.dot(y - t,X)
    return (dLdw, dLdb)
```

2 Explenation on Gradients

First we define our loss function as followed:

$L(w,b) = -\frac{1}{N}\sum_{n=1}^{N} t_n \log(y_n) + (1-t_n)\log(1-y_n)$.
Let $z_n = \mathbf{w}^T\mathbf{x}_n + b$
where we denote bold letter as a vector.
By the chain-rule:
$\frac{\partial L}{\partial w} = \sum_{n=1}^{N} \frac{\partial L}{\partial y_n}\cdot\frac{\partial y_n}{\partial z_n}\cdot\frac{\partial z_n}{\partial w}$
$\frac{\partial L}{\partial b} = \sum_{n=1}^{N} \frac{\partial L}{\partial y_n}\cdot\frac{\partial y_n}{\partial z_n}\cdot\frac{\partial z_n}{\partial b}$
where:
$\frac{\partial L}{\partial y_n} = -\frac{1}{N}\left(\frac{t_n}{y_n} - \frac{1-t_n}{1-y_n}\right) = \frac{y_n - t_n}{N\cdot y_n\cdot(1-y_n)}$,
$\frac{\partial y_n}{\partial z_n} = \sigma'(z_n) = \sigma(z_n)(1-\sigma(z_n)) = y_n(1-y_n)$
$\frac{\partial z_n}{\partial b} = 1$,

$\frac{\partial z_n}{\partial w} = \mathbf{x}_n$
These boils down to:
$\frac{\partial L}{\partial b} = \sum_{n=1}^{N} \frac{y_n - t_n}{N \cdot y_n \cdot (1-y_n)} \cdot y_n(1-y_n) \cdot 1 = \frac{1}{N} \sum_{n=1}^{N} y_n - t_n$
$\frac{\partial L}{\partial w} = \sum_{n=1}^{N} \frac{y_n - t_n}{N \cdot y_n \cdot (1-y_n)} \cdot y_n(1-y_n) \cdot \mathbf{x}_b = \frac{1}{N} \sum_{n=1}^{N} (y_n - t_n)\mathbf{x}_n$

2.0.1 Part (c) -- 7%

We can check that our derivative is implemented correctly using the finite difference rule. In 1D, the finite difference rule tells us that for small h, we should have

$$\frac{f(x+h) - f(x)}{h} \approx f'(x)$$

Show that $\frac{\partial \mathcal{L}}{\partial b}$ is implement correctly by comparing the result from `derivative_cost` with the empirical cost derivative computed using the above numerical approximation.

[ ]:
```
# Your code goes here
def compute_analytical_deriviate(w,b,t,X,h=1e-05):
  '''
  First we calculate the derivative with respect to the bias term
  '''
  # Calculate the predications
  y = pred(w, b, X)
  # Calculate the loss of those predications
  L = cost(y,t)
  y_new = pred(w,b+h,X)
  L_new = cost(y_new,t)

  # Calculate the analytic derivative
  deriviate_b =(L_new-L)/h

  '''
  Now we calculate the derivative with respect to the weights
  '''
  L_new = np.zeros(len(w))
  deriviate_w = np.zeros(len(w))
  for weight_idx in range(len(w)):
    # Compute cost of w[weight_idx] + h
    w_new = np.copy(w)
    w_new[weight_idx] = w_new[weight_idx] + h
    # Calculate the predications
    y_new = pred(w_new,b,X)
    # Calculate the loss of those predications
    L_new[weight_idx]=cost(y_new,t)
    deriviate_w[weight_idx] = (L_new[weight_idx]-L)/h

  return deriviate_w, deriviate_b
```

Now we are going to check our derivative using an example

[ ]:
```
w=np.zeros(90)
h=1e-05
b=1
t=np.array([1,1])
X=np.ones([2, 90])
y = pred(w, b, X)
dw,db = derivative_cost(X,y,t)
dw_analytic, db_analytic = compute_analytical_deriviate(w,b,t,X,h)
print("The bias analytical derivative results is -", db_analytic)
print("The bias algorithm derivative results is - ", db)
print(f"The error is {np.abs(db-db_analytic)}")
```

```
The bias analytical derivative results is - -0.26894043830827385
The bias algorithm derivative results is -  -0.2689414213699951
The error is 9.830617212491788e-07
```

2.0.2 Part (d) -- 7%

Show that $\frac{\partial \mathcal{L}}{\partial \mathbf{w}}$ is implement correctly.

[ ]:
```
print("The weights analytical derivative results is -", dw_analytic)
print("The weights algorithm derivative results is - ", dw)
print(f"The maximal error is {np.max(np.abs(dw-dw_analytic))}")
```

```
The weights analytical derivative results is - [-0.26894044 -0.26894044
-0.26894044 -0.26894044 -0.26894044 -0.26894044
 -0.26894044 -0.26894044 -0.26894044 -0.26894044 -0.26894044 -0.26894044
 -0.26894044 -0.26894044 -0.26894044 -0.26894044 -0.26894044 -0.26894044
 -0.26894044 -0.26894044 -0.26894044 -0.26894044 -0.26894044 -0.26894044
 -0.26894044 -0.26894044 -0.26894044 -0.26894044 -0.26894044 -0.26894044
 -0.26894044 -0.26894044 -0.26894044 -0.26894044 -0.26894044 -0.26894044
 -0.26894044 -0.26894044 -0.26894044 -0.26894044 -0.26894044 -0.26894044
 -0.26894044 -0.26894044 -0.26894044 -0.26894044 -0.26894044 -0.26894044
 -0.26894044 -0.26894044 -0.26894044 -0.26894044 -0.26894044 -0.26894044
 -0.26894044 -0.26894044 -0.26894044 -0.26894044 -0.26894044 -0.26894044
 -0.26894044 -0.26894044 -0.26894044 -0.26894044 -0.26894044 -0.26894044
 -0.26894044 -0.26894044 -0.26894044 -0.26894044 -0.26894044 -0.26894044
 -0.26894044 -0.26894044 -0.26894044 -0.26894044 -0.26894044 -0.26894044
 -0.26894044 -0.26894044 -0.26894044 -0.26894044 -0.26894044 -0.26894044
 -0.26894044 -0.26894044 -0.26894044 -0.26894044 -0.26894044 -0.26894044]
The weights algorithm derivative results is -  [-0.26894142 -0.26894142
-0.26894142 -0.26894142 -0.26894142 -0.26894142
 -0.26894142 -0.26894142 -0.26894142 -0.26894142 -0.26894142 -0.26894142
 -0.26894142 -0.26894142 -0.26894142 -0.26894142 -0.26894142 -0.26894142
 -0.26894142 -0.26894142 -0.26894142 -0.26894142 -0.26894142 -0.26894142
 -0.26894142 -0.26894142 -0.26894142 -0.26894142 -0.26894142 -0.26894142
 -0.26894142 -0.26894142 -0.26894142 -0.26894142 -0.26894142 -0.26894142
 -0.26894142 -0.26894142 -0.26894142 -0.26894142 -0.26894142 -0.26894142
 -0.26894142 -0.26894142 -0.26894142 -0.26894142 -0.26894142 -0.26894142
 -0.26894142 -0.26894142 -0.26894142 -0.26894142 -0.26894142 -0.26894142
 -0.26894142 -0.26894142 -0.26894142 -0.26894142 -0.26894142 -0.26894142
 -0.26894142 -0.26894142 -0.26894142 -0.26894142 -0.26894142 -0.26894142
 -0.26894142 -0.26894142 -0.26894142 -0.26894142 -0.26894142 -0.26894142
 -0.26894142 -0.26894142 -0.26894142 -0.26894142 -0.26894142 -0.26894142
 -0.26894142 -0.26894142 -0.26894142 -0.26894142 -0.26894142 -0.26894142
 -0.26894142 -0.26894142 -0.26894142 -0.26894142 -0.26894142 -0.26894142]
The maximal error is 9.830617212491788e-07
```

Now we will check over couple of real examples from the training set

```
[ ]: h = 1e-9;
     # First we initialize the weights by random numbers
     w=np.random.rand(90)
     b=np.random.rand(1)

     # Now we extract the actual samples from the dataset
     t = (train_ts[:1]).squeeze()
     X = train_norm_xs[:1]
     y = pred(w, b, X)

     dw,db = derivative_cost(X,y,t)
     dw_analytic, db_analytic = compute_analytical_deriviate(w,b,t,X,h)
     print("The analytical derivative bias results is -", db_analytic)
     print("The algorithm derivative bias results is - ", db)
     print(f"Bias Error is {np.abs(db-db_analytic)}")

     print("The analytical derivative weights results is -", dw_analytic)
     print("The algorithm derivative weights results is - ", dw)
     print(f"Maximal weights Error is {np.max(np.abs(dw-dw_analytic))}")
```

```
The analytical derivative bias results is - -0.9672760370449395
The algorithm derivative bias results is -  -0.9672759967800327
Bias Error is 4.0264906742137896e-08
The analytical derivative weights results is - [ 0.97880459  1.33712197
1.5150321   1.17287824 -0.1407332  -0.28086733
  0.99645137  1.66069425  0.34143222  2.53612864 -0.60000049  0.2660081
 -0.16358959  0.27339908 -0.18315438 -0.02620393  0.57133454  0.29290792
 -0.22191848 -0.26613023  0.32624214 -0.12356471  0.2247269  -0.1315108
  0.79538109  0.54556981  0.37574432  0.19421131  0.50898086  0.31249137
 -0.95565378 -0.54595706  0.15626922  0.11944934 -0.86183682 -0.96469277
 -0.30012703 -0.25347191 -0.2640399  -0.20129987  0.12283019  0.02585665
 -0.75847728  0.77280982  0.88499963 -0.2634799   0.25427704  0.86388496
 -0.74310602 -0.05215162  0.02074074 -0.18614754  0.64476735  0.68632078
 -0.79160367 -0.5105103  -0.51600768  0.62609251 -0.36517989  1.05336495
  0.91098018  0.16509771  0.39282799  0.36208059 -1.05262732  0.20108226
  0.01534106 -0.28936853  0.35263614 -0.85813934  0.51580074 -0.01067191
 -0.41914161  0.52804738 -0.2147611  -0.83647089  0.07340262 -1.70637771
 -0.25002356  0.29500091  0.20058133 -0.08705836  0.93793728  0.12571144
 -1.7977615  -0.31657921  0.64228134  0.73033668 -0.51113247  0.58821525]
The algorithm derivative weights results is -  [ 0.97880513  1.33712198
1.51503252  1.17287858 -0.14073285 -0.28086701
  0.99645136  1.66069465  0.34143218  2.53612882 -0.60000029  0.26600803
 -0.16358953  0.27339904 -0.18315388 -0.02620335  0.57133477  0.29290862
 -0.22191785 -0.26613003  0.3262425  -0.12356409  0.22472752 -0.13151021
  0.79538094  0.54556964  0.37574461  0.19421122  0.5089815   0.31249165
 -0.95565349 -0.54595737  0.15626943  0.11944995 -0.86183685 -0.96469248
 -0.30012698 -0.25347199 -0.26404002 -0.20129932  0.12282997  0.02585716
 -0.75847651  0.77280993  0.88499994 -0.26347972  0.25427681  0.86388554
 -0.74310507 -0.05215134  0.02074135 -0.18614762  0.64476718  0.68632109
 -0.79160365 -0.5105102  -0.51600786  0.62609255 -0.36517956  1.05336573
  0.91098042  0.16509771  0.39282848  0.3620809  -1.05262721  0.20108315
  0.01534155 -0.28936764  0.35263587 -0.8581389   0.51580074 -0.01067161
 -0.41914165  0.52804727 -0.21476141 -0.83647112  0.07340269 -1.70637775
 -0.25002302  0.29500102  0.20058141 -0.08705784  0.93793721  0.12571158
 -1.79776144 -0.31657967  0.64228073  0.73033717 -0.51113169  0.58821569]
Maximal weights Error is 9.504835127849276e-07
```

2.0.3 Part (e) -- 7%

Now that you have a gradient function that works, we can actually run gradient descent. Complete the following code that will run stochastic: gradient descent training:

```
[ ]: def run_gradient_descent(w0, b0, mu=0.1, batch_size=100, max_iters=100):
       """Return the values of (w, b) after running gradient descent for max_iters.
       We use:
         - train_norm_xs and train_ts as the training set
         - val_norm_xs and val_ts as the test set
         - mu as the learning rate
         - (w0, b0) as the initial values of (w, b)

       Precondition: np.shape(w0) == (90,)
                     type(b0) == float

       Postcondition: np.shape(w) == (90,)
                      type(b) == float
       """
       w = w0
       b = b0
       iter = 0
       val_loss = []

       while iter < max_iters:
         # shuffle the training set (there is code above for how to do this)
         new_indices = np.random.permutation(len(train_norm_xs))
         train_norm_xs_temp = train_norm_xs[new_indices]
```

```
    train_ts_temp = train_ts[new_indices]

    for i in range(0, len(train_norm_xs), batch_size): # iterate over each␣
↪minibatch
      # minibatch that we are working with:
      X = train_norm_xs_temp[i:(i + batch_size)]
      t = train_ts_temp[i:(i + batch_size), 0]

      # since len(train_norm_xs) does not divide batch_size evenly, we will␣
↪skip over
      # the "last" minibatch
      if np.shape(X)[0] != batch_size:
        continue

      # compute the prediction
      y = pred(w, b, X)
      # update w and b
      dw,db = derivative_cost(X,y,t)
      w -= mu*dw
      b -= mu*db

      # increment the iteration count
      iter += 1
      # compute and print the *validation* loss and accuracy
      if (iter % 10 == 0):
        # Calculate the predications over the validation data
        y_val = pred(w,b,val_norm_xs)
        # Calculate the cost over the predications
        val_cost = cost(y_val,val_ts.squeeze())
        val_acc = get_accuracy(y_val, val_ts)
        print("Iter %d. [Val Acc %.0f%%, Loss %f]" % (iter, val_acc * 100,␣
↪val_cost))
        val_loss.append(val_cost)
      if iter >= max_iters:
        break

      # Think what parameters you should return for further use
      # We choose to return in addition to w and b the validation loss in order␣
↪to evaluate our model training procedure over unseen data.
  return w, b, val_loss
```

2.0.4 Part (f) -- 7%

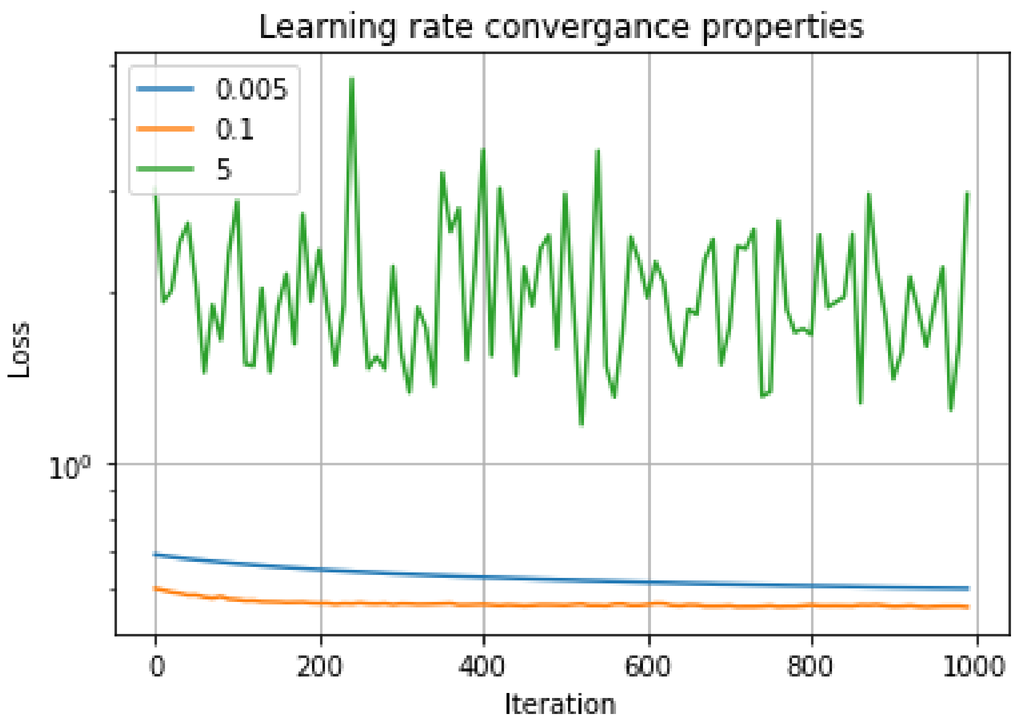
Call `run_gradient_descent` with the weights and biases all initialized to zero. Show that if the learning rate μ is too small, then convergence is slow. Also, show that if μ is too large, then the optimization algorirthm does not converge. The demonstration should be made using plots showing these effects.

[ ]:
```
w0 = np.zeros(90)
b0 = 0
Iterations = 1000
mus=[0.005,0.1,5]
for mu in mus:
  print("mu=",mu,":")
  w,b,losses=run_gradient_descent(w0,b0,mu, max_iters=Iterations)
  print("--------------------------")
  plt.semilogy(range(0,Iterations,10),losses)

plt.legend(mus,loc='best')
plt.xlabel("Iteration")
plt.ylabel("Loss")
plt.grid()
plt.title("Learning rate convergance properties")
```

```
mu= 0.005 :
Iter 10. [Val Acc 64%, Loss 0.689537]
Iter 20. [Val Acc 64%, Loss 0.686243]
Iter 30. [Val Acc 65%, Loss 0.683223]
Iter 40. [Val Acc 66%, Loss 0.680522]
Iter 50. [Val Acc 66%, Loss 0.677757]
Iter 60. [Val Acc 66%, Loss 0.675389]
Iter 70. [Val Acc 66%, Loss 0.673272]
Iter 80. [Val Acc 66%, Loss 0.670915]
Iter 90. [Val Acc 66%, Loss 0.668803]
Iter 100. [Val Acc 67%, Loss 0.666691]
Iter 110. [Val Acc 67%, Loss 0.664770]
Iter 120. [Val Acc 67%, Loss 0.663073]
Iter 130. [Val Acc 67%, Loss 0.661280]
Iter 140. [Val Acc 67%, Loss 0.659441]
Iter 150. [Val Acc 67%, Loss 0.657792]
Iter 160. [Val Acc 67%, Loss 0.656151]
Iter 170. [Val Acc 67%, Loss 0.654577]
Iter 180. [Val Acc 68%, Loss 0.653243]
Iter 190. [Val Acc 68%, Loss 0.651831]
Iter 200. [Val Acc 68%, Loss 0.650391]
Iter 210. [Val Acc 68%, Loss 0.649131]
Iter 220. [Val Acc 68%, Loss 0.647609]
Iter 230. [Val Acc 68%, Loss 0.646184]
Iter 240. [Val Acc 68%, Loss 0.645205]
Iter 250. [Val Acc 68%, Loss 0.644035]
Iter 260. [Val Acc 68%, Loss 0.642843]
Iter 270. [Val Acc 68%, Loss 0.641809]
Iter 280. [Val Acc 68%, Loss 0.640775]
Iter 290. [Val Acc 68%, Loss 0.639631]
Iter 300. [Val Acc 69%, Loss 0.638522]
Iter 310. [Val Acc 69%, Loss 0.637720]
Iter 320. [Val Acc 69%, Loss 0.636843]
Iter 330. [Val Acc 69%, Loss 0.635823]
Iter 340. [Val Acc 69%, Loss 0.635013]
Iter 350. [Val Acc 69%, Loss 0.634134]
Iter 360. [Val Acc 69%, Loss 0.633340]
Iter 370. [Val Acc 69%, Loss 0.632425]
Iter 380. [Val Acc 69%, Loss 0.631660]
Iter 390. [Val Acc 69%, Loss 0.630979]
Iter 400. [Val Acc 69%, Loss 0.630316]
Iter 410. [Val Acc 69%, Loss 0.629490]
Iter 420. [Val Acc 69%, Loss 0.628746]
Iter 430. [Val Acc 69%, Loss 0.628168]
Iter 440. [Val Acc 69%, Loss 0.627399]
Iter 450. [Val Acc 69%, Loss 0.626611]
Iter 460. [Val Acc 69%, Loss 0.625946]
Iter 470. [Val Acc 69%, Loss 0.625162]
Iter 480. [Val Acc 69%, Loss 0.624443]
Iter 490. [Val Acc 69%, Loss 0.623948]
Iter 500. [Val Acc 69%, Loss 0.623216]
Iter 510. [Val Acc 69%, Loss 0.622644]
Iter 520. [Val Acc 70%, Loss 0.622084]
Iter 530. [Val Acc 70%, Loss 0.621376]
Iter 540. [Val Acc 70%, Loss 0.620889]
Iter 550. [Val Acc 70%, Loss 0.620391]
Iter 560. [Val Acc 70%, Loss 0.619677]
Iter 570. [Val Acc 70%, Loss 0.619092]
Iter 580. [Val Acc 70%, Loss 0.618571]
Iter 590. [Val Acc 70%, Loss 0.617980]
Iter 600. [Val Acc 70%, Loss 0.617424]
Iter 610. [Val Acc 70%, Loss 0.616836]
Iter 620. [Val Acc 70%, Loss 0.616278]
Iter 630. [Val Acc 70%, Loss 0.615821]
Iter 640. [Val Acc 70%, Loss 0.615303]
Iter 650. [Val Acc 70%, Loss 0.614726]
Iter 660. [Val Acc 70%, Loss 0.614270]
Iter 670. [Val Acc 70%, Loss 0.613770]
Iter 680. [Val Acc 70%, Loss 0.613442]
Iter 690. [Val Acc 70%, Loss 0.612879]
Iter 700. [Val Acc 70%, Loss 0.612325]
Iter 710. [Val Acc 70%, Loss 0.612043]
Iter 720. [Val Acc 70%, Loss 0.611601]
Iter 730. [Val Acc 70%, Loss 0.611160]
Iter 740. [Val Acc 70%, Loss 0.610755]
Iter 750. [Val Acc 70%, Loss 0.610356]
Iter 760. [Val Acc 70%, Loss 0.609983]
Iter 770. [Val Acc 70%, Loss 0.609521]
Iter 780. [Val Acc 70%, Loss 0.609124]
Iter 790. [Val Acc 70%, Loss 0.608796]
Iter 800. [Val Acc 70%, Loss 0.608408]
Iter 810. [Val Acc 70%, Loss 0.608064]
Iter 820. [Val Acc 70%, Loss 0.607669]
Iter 830. [Val Acc 70%, Loss 0.607306]
Iter 840. [Val Acc 70%, Loss 0.606962]
Iter 850. [Val Acc 70%, Loss 0.606637]
Iter 860. [Val Acc 70%, Loss 0.606213]
Iter 870. [Val Acc 70%, Loss 0.605830]
Iter 880. [Val Acc 70%, Loss 0.605357]
Iter 890. [Val Acc 70%, Loss 0.605005]
Iter 900. [Val Acc 71%, Loss 0.604605]
Iter 910. [Val Acc 71%, Loss 0.604260]
Iter 920. [Val Acc 71%, Loss 0.604015]
Iter 930. [Val Acc 71%, Loss 0.603699]
Iter 940. [Val Acc 71%, Loss 0.603412]
Iter 950. [Val Acc 71%, Loss 0.603159]
Iter 960. [Val Acc 71%, Loss 0.602894]
Iter 970. [Val Acc 71%, Loss 0.602661]
Iter 980. [Val Acc 71%, Loss 0.602427]
Iter 990. [Val Acc 71%, Loss 0.602187]
Iter 1000. [Val Acc 71%, Loss 0.601942]
-------------------------
mu= 0.1 :
Iter 10. [Val Acc 71%, Loss 0.601829]
Iter 20. [Val Acc 71%, Loss 0.597514]
Iter 30. [Val Acc 71%, Loss 0.592957]
Iter 40. [Val Acc 71%, Loss 0.590058]
Iter 50. [Val Acc 72%, Loss 0.585832]
Iter 60. [Val Acc 72%, Loss 0.586104]
Iter 70. [Val Acc 72%, Loss 0.581339]
Iter 80. [Val Acc 72%, Loss 0.578768]
Iter 90. [Val Acc 71%, Loss 0.582677]
Iter 100. [Val Acc 72%, Loss 0.576505]
Iter 110. [Val Acc 72%, Loss 0.575099]
Iter 120. [Val Acc 73%, Loss 0.572667]
Iter 130. [Val Acc 73%, Loss 0.572912]
Iter 140. [Val Acc 73%, Loss 0.570543]
Iter 150. [Val Acc 73%, Loss 0.570110]
Iter 160. [Val Acc 73%, Loss 0.569673]
Iter 170. [Val Acc 73%, Loss 0.568572]
Iter 180. [Val Acc 73%, Loss 0.568890]
Iter 190. [Val Acc 73%, Loss 0.569118]
Iter 200. [Val Acc 73%, Loss 0.567459]
Iter 210. [Val Acc 73%, Loss 0.566941]
Iter 220. [Val Acc 73%, Loss 0.567065]
Iter 230. [Val Acc 73%, Loss 0.564534]
Iter 240. [Val Acc 73%, Loss 0.566419]
Iter 250. [Val Acc 73%, Loss 0.565836]
Iter 260. [Val Acc 73%, Loss 0.567727]
Iter 270. [Val Acc 73%, Loss 0.565815]
Iter 280. [Val Acc 73%, Loss 0.565426]
Iter 290. [Val Acc 73%, Loss 0.566482]
Iter 300. [Val Acc 73%, Loss 0.563875]
Iter 310. [Val Acc 73%, Loss 0.565938]
Iter 320. [Val Acc 73%, Loss 0.565502]
Iter 330. [Val Acc 73%, Loss 0.564350]
Iter 340. [Val Acc 73%, Loss 0.564619]
Iter 350. [Val Acc 73%, Loss 0.565174]
Iter 360. [Val Acc 73%, Loss 0.565855]
Iter 370. [Val Acc 73%, Loss 0.566714]
Iter 380. [Val Acc 73%, Loss 0.563141]
Iter 390. [Val Acc 73%, Loss 0.563240]
Iter 400. [Val Acc 73%, Loss 0.563809]
Iter 410. [Val Acc 73%, Loss 0.564465]
Iter 420. [Val Acc 73%, Loss 0.563139]
Iter 430. [Val Acc 73%, Loss 0.561894]
Iter 440. [Val Acc 73%, Loss 0.563165]
Iter 450. [Val Acc 73%, Loss 0.562376]
Iter 460. [Val Acc 73%, Loss 0.561198]
Iter 470. [Val Acc 73%, Loss 0.562118]
Iter 480. [Val Acc 73%, Loss 0.563457]
Iter 490. [Val Acc 73%, Loss 0.563346]
Iter 500. [Val Acc 73%, Loss 0.562997]
Iter 510. [Val Acc 73%, Loss 0.562334]
Iter 520. [Val Acc 73%, Loss 0.563949]
Iter 530. [Val Acc 73%, Loss 0.565185]
Iter 540. [Val Acc 73%, Loss 0.561996]
Iter 550. [Val Acc 73%, Loss 0.562082]
Iter 560. [Val Acc 73%, Loss 0.561230]
Iter 570. [Val Acc 73%, Loss 0.564939]
Iter 580. [Val Acc 73%, Loss 0.565364]
Iter 590. [Val Acc 73%, Loss 0.562021]
Iter 600. [Val Acc 73%, Loss 0.562121]
Iter 610. [Val Acc 73%, Loss 0.563909]
Iter 620. [Val Acc 73%, Loss 0.566711]
Iter 630. [Val Acc 73%, Loss 0.566379]
Iter 640. [Val Acc 73%, Loss 0.562145]
Iter 650. [Val Acc 73%, Loss 0.561149]
Iter 660. [Val Acc 73%, Loss 0.563261]
Iter 670. [Val Acc 73%, Loss 0.562966]
Iter 680. [Val Acc 73%, Loss 0.560195]
Iter 690. [Val Acc 73%, Loss 0.560050]
Iter 700. [Val Acc 73%, Loss 0.560278]
Iter 710. [Val Acc 73%, Loss 0.561291]
Iter 720. [Val Acc 74%, Loss 0.558807]
Iter 730. [Val Acc 73%, Loss 0.559066]
Iter 740. [Val Acc 73%, Loss 0.559120]
Iter 750. [Val Acc 73%, Loss 0.560010]
Iter 760. [Val Acc 73%, Loss 0.561255]
Iter 770. [Val Acc 73%, Loss 0.558800]
Iter 780. [Val Acc 73%, Loss 0.559677]
Iter 790. [Val Acc 73%, Loss 0.559619]
Iter 800. [Val Acc 73%, Loss 0.561323]
Iter 810. [Val Acc 73%, Loss 0.563050]
Iter 820. [Val Acc 73%, Loss 0.561049]
Iter 830. [Val Acc 73%, Loss 0.560999]
Iter 840. [Val Acc 73%, Loss 0.561243]
Iter 850. [Val Acc 73%, Loss 0.560951]
Iter 860. [Val Acc 73%, Loss 0.560676]
Iter 870. [Val Acc 73%, Loss 0.563061]
Iter 880. [Val Acc 73%, Loss 0.562676]
Iter 890. [Val Acc 73%, Loss 0.563640]
Iter 900. [Val Acc 73%, Loss 0.561361]
Iter 910. [Val Acc 73%, Loss 0.559478]
Iter 920. [Val Acc 73%, Loss 0.559973]
Iter 930. [Val Acc 73%, Loss 0.561349]
Iter 940. [Val Acc 73%, Loss 0.559364]
Iter 950. [Val Acc 73%, Loss 0.558207]
Iter 960. [Val Acc 73%, Loss 0.559179]
Iter 970. [Val Acc 74%, Loss 0.559837]
Iter 980. [Val Acc 74%, Loss 0.559884]
Iter 990. [Val Acc 73%, Loss 0.560026]
Iter 1000. [Val Acc 74%, Loss 0.558236]
-------------------------
mu= 5 :
Iter 10. [Val Acc 59%, Loss 3.023958]
Iter 20. [Val Acc 57%, Loss 1.914329]
Iter 30. [Val Acc 62%, Loss 2.002755]
Iter 40. [Val Acc 64%, Loss 2.435203]
Iter 50. [Val Acc 63%, Loss 2.626566]
Iter 60. [Val Acc 66%, Loss 2.039638]
Iter 70. [Val Acc 67%, Loss 1.439940]
Iter 80. [Val Acc 68%, Loss 1.891332]
Iter 90. [Val Acc 65%, Loss 1.640083]
Iter 100. [Val Acc 61%, Loss 2.337751]
Iter 110. [Val Acc 59%, Loss 2.873740]
Iter 120. [Val Acc 67%, Loss 1.488335]
Iter 130. [Val Acc 67%, Loss 1.477086]
Iter 140. [Val Acc 63%, Loss 2.021669]
Iter 150. [Val Acc 66%, Loss 1.442532]
Iter 160. [Val Acc 65%, Loss 1.874325]
Iter 170. [Val Acc 61%, Loss 2.141212]
Iter 180. [Val Acc 66%, Loss 1.613201]
Iter 190. [Val Acc 62%, Loss 2.725409]
Iter 200. [Val Acc 62%, Loss 1.916633]
Iter 210. [Val Acc 61%, Loss 2.360604]
Iter 220. [Val Acc 62%, Loss 1.854672]
Iter 230. [Val Acc 66%, Loss 1.481538]
Iter 240. [Val Acc 64%, Loss 1.888021]
Iter 250. [Val Acc 57%, Loss 4.709104]
Iter 260. [Val Acc 64%, Loss 2.027701]
Iter 270. [Val Acc 67%, Loss 1.461865]
Iter 280. [Val Acc 66%, Loss 1.532466]
Iter 290. [Val Acc 67%, Loss 1.459057]
Iter 300. [Val Acc 62%, Loss 2.208295]
Iter 310. [Val Acc 69%, Loss 1.556971]
Iter 320. [Val Acc 68%, Loss 1.326834]
Iter 330. [Val Acc 67%, Loss 1.871055]
Iter 340. [Val Acc 64%, Loss 1.723846]
Iter 350. [Val Acc 66%, Loss 1.364312]
Iter 360. [Val Acc 62%, Loss 3.223462]
Iter 370. [Val Acc 62%, Loss 2.537601]
Iter 380. [Val Acc 57%, Loss 2.789633]
Iter 390. [Val Acc 67%, Loss 1.511727]
Iter 400. [Val Acc 65%, Loss 2.208063]
Iter 410. [Val Acc 56%, Loss 3.532175]
Iter 420. [Val Acc 67%, Loss 1.538770]
Iter 430. [Val Acc 57%, Loss 3.027161]
Iter 440. [Val Acc 62%, Loss 2.283982]
Iter 450. [Val Acc 67%, Loss 1.421928]
Iter 460. [Val Acc 63%, Loss 2.200051]
Iter 470. [Val Acc 64%, Loss 1.882144]
Iter 480. [Val Acc 57%, Loss 2.370721]
Iter 490. [Val Acc 57%, Loss 2.502125]
Iter 500. [Val Acc 67%, Loss 1.588496]
Iter 510. [Val Acc 61%, Loss 2.959032]
Iter 520. [Val Acc 68%, Loss 1.879302]
Iter 530. [Val Acc 68%, Loss 1.164676]
Iter 540. [Val Acc 64%, Loss 1.824828]
Iter 550. [Val Acc 57%, Loss 3.520049]
Iter 560. [Val Acc 69%, Loss 1.475708]
Iter 570. [Val Acc 68%, Loss 1.303260]
Iter 580. [Val Acc 68%, Loss 1.693122]
Iter 590. [Val Acc 63%, Loss 2.487929]
Iter 600. [Val Acc 63%, Loss 2.252570]
Iter 610. [Val Acc 62%, Loss 1.945547]
Iter 620. [Val Acc 60%, Loss 2.248732]
Iter 630. [Val Acc 63%, Loss 2.068087]
Iter 640. [Val Acc 66%, Loss 1.637215]
Iter 650. [Val Acc 68%, Loss 1.478781]
Iter 660. [Val Acc 65%, Loss 1.856137]
Iter 670. [Val Acc 66%, Loss 1.817354]
Iter 680. [Val Acc 62%, Loss 2.266390]
Iter 690. [Val Acc 59%, Loss 2.458833]
Iter 700. [Val Acc 67%, Loss 1.485949]
Iter 710. [Val Acc 65%, Loss 1.720554]
Iter 720. [Val Acc 64%, Loss 2.393605]
Iter 730. [Val Acc 63%, Loss 2.371198]
Iter 740. [Val Acc 62%, Loss 2.561722]
Iter 750. [Val Acc 68%, Loss 1.306933]
Iter 760. [Val Acc 69%, Loss 1.331141]
Iter 770. [Val Acc 64%, Loss 2.654016]
Iter 780. [Val Acc 67%, Loss 1.854800]
Iter 790. [Val Acc 68%, Loss 1.690965]
Iter 800. [Val Acc 65%, Loss 1.717231]
Iter 810. [Val Acc 64%, Loss 1.676793]
Iter 820. [Val Acc 62%, Loss 2.508890]
Iter 830. [Val Acc 63%, Loss 1.871474]
Iter 840. [Val Acc 62%, Loss 1.912280]
Iter 850. [Val Acc 64%, Loss 1.950877]
Iter 860. [Val Acc 63%, Loss 2.517294]
Iter 870. [Val Acc 70%, Loss 1.272444]
Iter 880. [Val Acc 61%, Loss 2.958198]
Iter 890. [Val Acc 61%, Loss 2.191611]
Iter 900. [Val Acc 62%, Loss 1.816850]
Iter 910. [Val Acc 68%, Loss 1.400787]
Iter 920. [Val Acc 68%, Loss 1.559446]
Iter 930. [Val Acc 65%, Loss 2.116104]
Iter 940. [Val Acc 65%, Loss 1.849257]
Iter 950. [Val Acc 67%, Loss 1.597749]
Iter 960. [Val Acc 63%, Loss 1.887086]
Iter 970. [Val Acc 65%, Loss 2.198277]
Iter 980. [Val Acc 70%, Loss 1.238432]
Iter 990. [Val Acc 68%, Loss 1.616702]
Iter 1000. [Val Acc 59%, Loss 2.957832]
-------------------------
```

[ ]:
```
Text(0.5, 1.0, 'Learning rate convergance properties')
```



We conclude from the figure above that for a small learning rate, the algorithm may be capable to converge but very slowly due to the small changes in the parameters space. On the other hand, we see that by using too large learning rate, the algorithm is incapable to converge due to the large changes in the parameters space. Thus, we conclude from the empirical results that choosing a learning rate somewhere in between those learning rates leads to the best performance. We explain this phenomenon by the fact that the algorithm can change its parameters efficiently towards a good optimum. Meaning that we have found a good trade-off between a small enough step size that allows convergence but not too small such that it leads to faster convergence rate to a good enough optimum.

2.0.5 Part (g) -- 7%

Find the optimial value of **w** and b using your code. Explain how you chose the learning rate μ and the batch size. Show plots demostrating good and bad behaviours.

[ ]:
```
'''
Hyperparameters search
'''

mus = [0.01, 0.1, 0.5, 1]
batch_sizes = [1, 64, 1024, 50000]
Iterations = 1000
loss_results = np.zeros((len(mus),len(batch_sizes)))
w_history = np.zeros((len(mus),len(batch_sizes), 90))
b_history = np.zeros((len(mus),len(batch_sizes), 1))
```

```
w0 = np.random.randn(90)
b0 = np.random.randn(1)[0]

for mu_idx,mu in enumerate(mus):
  for minibatch_idx, minibatch_size in enumerate(batch_sizes):
    print("mu=",mu, 'batch_size=', minibatch_size, ':')
    w,b,losses=run_gradient_descent(w0,b0,mu, max_iters=Iterations)
    print("-------------------------")
    loss_results[mu_idx][minibatch_idx] = losses[-1]
    w_history[mu_idx][minibatch_idx] = w
    b_history[mu_idx][minibatch_idx] = b

best_mu_idx, best_batch_size_idx = np.unravel_index(np.argmin(loss_results,␣
 ↪axis=None), loss_results.shape)
print("optimal mu=",mus[best_mu_idx], 'optimal batch_size=',␣
 ↪batch_sizes[best_batch_size_idx])
'''
Save the optimal weight and bias
'''
opt_w = w_history[best_mu_idx][best_batch_size_idx]
opt_b = b_history[best_mu_idx][best_batch_size_idx]
```

```
mu= 0.01 batch_size= 1 :
Iter 10. [Val Acc 47%, Loss 2.747708]
Iter 20. [Val Acc 47%, Loss 2.714000]
Iter 30. [Val Acc 47%, Loss 2.686253]
Iter 40. [Val Acc 47%, Loss 2.659955]
Iter 50. [Val Acc 47%, Loss 2.633081]
Iter 60. [Val Acc 47%, Loss 2.603889]
Iter 70. [Val Acc 48%, Loss 2.580692]
Iter 80. [Val Acc 48%, Loss 2.557785]
Iter 90. [Val Acc 48%, Loss 2.530571]
Iter 100. [Val Acc 48%, Loss 2.503728]
Iter 110. [Val Acc 48%, Loss 2.479285]
Iter 120. [Val Acc 48%, Loss 2.455511]
Iter 130. [Val Acc 48%, Loss 2.436268]
Iter 140. [Val Acc 48%, Loss 2.415122]
Iter 150. [Val Acc 48%, Loss 2.394333]
Iter 160. [Val Acc 48%, Loss 2.377478]
Iter 170. [Val Acc 48%, Loss 2.357796]
Iter 180. [Val Acc 48%, Loss 2.338882]
Iter 190. [Val Acc 48%, Loss 2.319985]
Iter 200. [Val Acc 48%, Loss 2.298555]
Iter 210. [Val Acc 48%, Loss 2.279912]
Iter 220. [Val Acc 49%, Loss 2.262655]
Iter 230. [Val Acc 49%, Loss 2.244858]
Iter 240. [Val Acc 49%, Loss 2.228405]
Iter 250. [Val Acc 49%, Loss 2.213221]
Iter 260. [Val Acc 49%, Loss 2.195344]
Iter 270. [Val Acc 49%, Loss 2.180319]
Iter 280. [Val Acc 49%, Loss 2.164565]
Iter 290. [Val Acc 49%, Loss 2.145138]
Iter 300. [Val Acc 49%, Loss 2.128512]
Iter 310. [Val Acc 49%, Loss 2.111651]
Iter 320. [Val Acc 49%, Loss 2.095608]
Iter 330. [Val Acc 49%, Loss 2.079349]
Iter 340. [Val Acc 50%, Loss 2.063636]
Iter 350. [Val Acc 50%, Loss 2.049249]
Iter 360. [Val Acc 50%, Loss 2.034857]
Iter 370. [Val Acc 50%, Loss 2.018879]
Iter 380. [Val Acc 50%, Loss 2.003994]
Iter 390. [Val Acc 50%, Loss 1.988755]
Iter 400. [Val Acc 50%, Loss 1.975895]
Iter 410. [Val Acc 50%, Loss 1.960965]
Iter 420. [Val Acc 50%, Loss 1.946616]
Iter 430. [Val Acc 50%, Loss 1.931607]
Iter 440. [Val Acc 50%, Loss 1.917479]
Iter 450. [Val Acc 50%, Loss 1.902136]
Iter 460. [Val Acc 50%, Loss 1.888529]
Iter 470. [Val Acc 51%, Loss 1.877311]
Iter 480. [Val Acc 51%, Loss 1.865819]
Iter 490. [Val Acc 51%, Loss 1.851953]
Iter 500. [Val Acc 51%, Loss 1.839929]
Iter 510. [Val Acc 51%, Loss 1.826131]
Iter 520. [Val Acc 51%, Loss 1.814031]
Iter 530. [Val Acc 51%, Loss 1.801829]
Iter 540. [Val Acc 51%, Loss 1.788531]
Iter 550. [Val Acc 51%, Loss 1.775909]
Iter 560. [Val Acc 51%, Loss 1.763521]
Iter 570. [Val Acc 51%, Loss 1.751194]
Iter 580. [Val Acc 51%, Loss 1.739099]
Iter 590. [Val Acc 52%, Loss 1.727596]
Iter 600. [Val Acc 52%, Loss 1.716846]
Iter 610. [Val Acc 52%, Loss 1.705644]
Iter 620. [Val Acc 52%, Loss 1.694009]
Iter 630. [Val Acc 52%, Loss 1.683048]
Iter 640. [Val Acc 52%, Loss 1.672278]
Iter 650. [Val Acc 52%, Loss 1.660642]
Iter 660. [Val Acc 52%, Loss 1.649790]
Iter 670. [Val Acc 52%, Loss 1.639040]
Iter 680. [Val Acc 53%, Loss 1.626574]
Iter 690. [Val Acc 53%, Loss 1.616777]
Iter 700. [Val Acc 53%, Loss 1.605414]
Iter 710. [Val Acc 53%, Loss 1.594303]
Iter 720. [Val Acc 53%, Loss 1.585565]
Iter 730. [Val Acc 53%, Loss 1.576315]
Iter 740. [Val Acc 53%, Loss 1.566632]
Iter 750. [Val Acc 53%, Loss 1.557272]
Iter 760. [Val Acc 53%, Loss 1.547641]
Iter 770. [Val Acc 53%, Loss 1.539211]
Iter 780. [Val Acc 53%, Loss 1.530813]
Iter 790. [Val Acc 54%, Loss 1.521222]
Iter 800. [Val Acc 54%, Loss 1.512906]
Iter 810. [Val Acc 54%, Loss 1.503275]
Iter 820. [Val Acc 54%, Loss 1.494597]
Iter 830. [Val Acc 54%, Loss 1.485597]
Iter 840. [Val Acc 54%, Loss 1.476089]
Iter 850. [Val Acc 54%, Loss 1.468392]
Iter 860. [Val Acc 54%, Loss 1.460788]
Iter 870. [Val Acc 54%, Loss 1.452696]
Iter 880. [Val Acc 54%, Loss 1.443291]
Iter 890. [Val Acc 54%, Loss 1.436108]
Iter 900. [Val Acc 54%, Loss 1.426818]
Iter 910. [Val Acc 54%, Loss 1.418355]
Iter 920. [Val Acc 55%, Loss 1.410911]
Iter 930. [Val Acc 55%, Loss 1.402065]
Iter 940. [Val Acc 55%, Loss 1.394395]
Iter 950. [Val Acc 55%, Loss 1.386117]
Iter 960. [Val Acc 55%, Loss 1.377559]
Iter 970. [Val Acc 55%, Loss 1.370999]
Iter 980. [Val Acc 55%, Loss 1.363807]
Iter 990. [Val Acc 55%, Loss 1.356797]
Iter 1000. [Val Acc 55%, Loss 1.349479]
-------------------------
mu= 0.01 batch_size= 64 :
Iter 10. [Val Acc 55%, Loss 1.345617]
Iter 20. [Val Acc 56%, Loss 1.338529]
Iter 30. [Val Acc 56%, Loss 1.331585]
Iter 40. [Val Acc 56%, Loss 1.325503]
Iter 50. [Val Acc 56%, Loss 1.318606]
Iter 60. [Val Acc 56%, Loss 1.310718]
Iter 70. [Val Acc 56%, Loss 1.304447]
Iter 80. [Val Acc 56%, Loss 1.297898]
Iter 90. [Val Acc 56%, Loss 1.290197]
Iter 100. [Val Acc 56%, Loss 1.283876]
Iter 110. [Val Acc 56%, Loss 1.277738]
Iter 120. [Val Acc 56%, Loss 1.271026]
Iter 130. [Val Acc 56%, Loss 1.263781]
Iter 140. [Val Acc 56%, Loss 1.257526]
Iter 150. [Val Acc 57%, Loss 1.251464]
Iter 160. [Val Acc 57%, Loss 1.245631]
Iter 170. [Val Acc 57%, Loss 1.239308]
Iter 180. [Val Acc 57%, Loss 1.232742]
Iter 190. [Val Acc 57%, Loss 1.227313]
Iter 200. [Val Acc 57%, Loss 1.221064]
Iter 210. [Val Acc 57%, Loss 1.214861]
Iter 220. [Val Acc 57%, Loss 1.208799]
Iter 230. [Val Acc 57%, Loss 1.202713]
Iter 240. [Val Acc 57%, Loss 1.197273]
Iter 250. [Val Acc 57%, Loss 1.191149]
Iter 260. [Val Acc 57%, Loss 1.185029]
Iter 270. [Val Acc 57%, Loss 1.179632]
Iter 280. [Val Acc 57%, Loss 1.174148]
Iter 290. [Val Acc 57%, Loss 1.168859]
Iter 300. [Val Acc 57%, Loss 1.163904]
Iter 310. [Val Acc 58%, Loss 1.158641]
Iter 320. [Val Acc 58%, Loss 1.153226]
Iter 330. [Val Acc 58%, Loss 1.148096]
Iter 340. [Val Acc 58%, Loss 1.142611]
Iter 350. [Val Acc 58%, Loss 1.137139]
Iter 360. [Val Acc 58%, Loss 1.132120]
Iter 370. [Val Acc 58%, Loss 1.126368]
Iter 380. [Val Acc 58%, Loss 1.121926]
Iter 390. [Val Acc 58%, Loss 1.116211]
Iter 400. [Val Acc 58%, Loss 1.111316]
Iter 410. [Val Acc 58%, Loss 1.107111]
Iter 420. [Val Acc 58%, Loss 1.102484]
Iter 430. [Val Acc 58%, Loss 1.098378]
Iter 440. [Val Acc 58%, Loss 1.094737]
Iter 450. [Val Acc 58%, Loss 1.090213]
Iter 460. [Val Acc 59%, Loss 1.085753]
Iter 470. [Val Acc 59%, Loss 1.081230]
Iter 480. [Val Acc 59%, Loss 1.076412]
Iter 490. [Val Acc 59%, Loss 1.072144]
Iter 500. [Val Acc 59%, Loss 1.067320]
Iter 510. [Val Acc 59%, Loss 1.061990]
Iter 520. [Val Acc 59%, Loss 1.057249]
Iter 530. [Val Acc 59%, Loss 1.053846]
Iter 540. [Val Acc 59%, Loss 1.049935]
Iter 550. [Val Acc 59%, Loss 1.046121]
Iter 560. [Val Acc 59%, Loss 1.041967]
Iter 570. [Val Acc 59%, Loss 1.038177]
Iter 580. [Val Acc 59%, Loss 1.033472]
Iter 590. [Val Acc 59%, Loss 1.029379]
Iter 600. [Val Acc 59%, Loss 1.024871]
Iter 610. [Val Acc 59%, Loss 1.020544]
Iter 620. [Val Acc 59%, Loss 1.017094]
Iter 630. [Val Acc 60%, Loss 1.013561]
Iter 640. [Val Acc 60%, Loss 1.009934]
Iter 650. [Val Acc 60%, Loss 1.005945]
Iter 660. [Val Acc 60%, Loss 1.002079]
Iter 670. [Val Acc 60%, Loss 0.998660]
Iter 680. [Val Acc 60%, Loss 0.994886]
Iter 690. [Val Acc 60%, Loss 0.991315]
Iter 700. [Val Acc 60%, Loss 0.987764]
Iter 710. [Val Acc 60%, Loss 0.984638]
Iter 720. [Val Acc 60%, Loss 0.981326]
Iter 730. [Val Acc 60%, Loss 0.977929]
Iter 740. [Val Acc 60%, Loss 0.974272]
Iter 750. [Val Acc 60%, Loss 0.971192]
Iter 760. [Val Acc 60%, Loss 0.968324]
Iter 770. [Val Acc 60%, Loss 0.965537]
Iter 780. [Val Acc 60%, Loss 0.962614]
Iter 790. [Val Acc 61%, Loss 0.959136]
Iter 800. [Val Acc 61%, Loss 0.956208]
Iter 810. [Val Acc 61%, Loss 0.953530]
Iter 820. [Val Acc 61%, Loss 0.950446]
Iter 830. [Val Acc 61%, Loss 0.946679]
Iter 840. [Val Acc 61%, Loss 0.943529]
Iter 850. [Val Acc 61%, Loss 0.940725]
Iter 860. [Val Acc 61%, Loss 0.937726]
Iter 870. [Val Acc 61%, Loss 0.934797]
Iter 880. [Val Acc 61%, Loss 0.931442]
Iter 890. [Val Acc 61%, Loss 0.928389]
Iter 900. [Val Acc 61%, Loss 0.925764]
Iter 910. [Val Acc 61%, Loss 0.922628]
Iter 920. [Val Acc 61%, Loss 0.919816]
Iter 930. [Val Acc 61%, Loss 0.917270]
Iter 940. [Val Acc 62%, Loss 0.914550]
Iter 950. [Val Acc 62%, Loss 0.911570]
Iter 960. [Val Acc 62%, Loss 0.909385]
Iter 970. [Val Acc 62%, Loss 0.906585]
Iter 980. [Val Acc 62%, Loss 0.904106]
Iter 990. [Val Acc 62%, Loss 0.901269]
Iter 1000. [Val Acc 62%, Loss 0.898525]
-------------------------
mu= 0.01 batch_size= 1024 :
Iter 10. [Val Acc 62%, Loss 0.906328]
Iter 20. [Val Acc 62%, Loss 0.903532]
Iter 30. [Val Acc 62%, Loss 0.901044]
Iter 40. [Val Acc 62%, Loss 0.898690]
Iter 50. [Val Acc 62%, Loss 0.896345]
Iter 60. [Val Acc 62%, Loss 0.893715]
Iter 70. [Val Acc 62%, Loss 0.891497]
Iter 80. [Val Acc 62%, Loss 0.888453]
Iter 90. [Val Acc 62%, Loss 0.886079]
Iter 100. [Val Acc 62%, Loss 0.883580]
Iter 110. [Val Acc 62%, Loss 0.880946]
Iter 120. [Val Acc 62%, Loss 0.878768]
Iter 130. [Val Acc 62%, Loss 0.876273]
Iter 140. [Val Acc 62%, Loss 0.874109]
Iter 150. [Val Acc 63%, Loss 0.871913]
Iter 160. [Val Acc 63%, Loss 0.869762]
Iter 170. [Val Acc 63%, Loss 0.867458]
Iter 180. [Val Acc 63%, Loss 0.865235]
Iter 190. [Val Acc 63%, Loss 0.863043]
Iter 200. [Val Acc 63%, Loss 0.860969]
Iter 210. [Val Acc 63%, Loss 0.859043]
Iter 220. [Val Acc 63%, Loss 0.856785]
Iter 230. [Val Acc 63%, Loss 0.854760]
Iter 240. [Val Acc 63%, Loss 0.852633]
Iter 250. [Val Acc 63%, Loss 0.850425]
Iter 260. [Val Acc 63%, Loss 0.848362]
Iter 270. [Val Acc 63%, Loss 0.846445]
Iter 280. [Val Acc 63%, Loss 0.844571]
Iter 290. [Val Acc 63%, Loss 0.842505]
Iter 300. [Val Acc 63%, Loss 0.840632]
Iter 310. [Val Acc 63%, Loss 0.838513]
Iter 320. [Val Acc 63%, Loss 0.836432]
Iter 330. [Val Acc 63%, Loss 0.834393]
Iter 340. [Val Acc 63%, Loss 0.832571]
Iter 350. [Val Acc 64%, Loss 0.830582]
Iter 360. [Val Acc 64%, Loss 0.828517]
Iter 370. [Val Acc 64%, Loss 0.826750]
Iter 380. [Val Acc 64%, Loss 0.824939]
Iter 390. [Val Acc 64%, Loss 0.823122]
Iter 400. [Val Acc 64%, Loss 0.821391]
Iter 410. [Val Acc 64%, Loss 0.819657]
Iter 420. [Val Acc 64%, Loss 0.818024]
Iter 430. [Val Acc 64%, Loss 0.815976]
Iter 440. [Val Acc 64%, Loss 0.814484]
Iter 450. [Val Acc 64%, Loss 0.813169]
Iter 460. [Val Acc 64%, Loss 0.811369]
Iter 470. [Val Acc 64%, Loss 0.809472]
Iter 480. [Val Acc 64%, Loss 0.807631]
Iter 490. [Val Acc 64%, Loss 0.805938]
Iter 500. [Val Acc 64%, Loss 0.804531]
Iter 510. [Val Acc 64%, Loss 0.803300]
Iter 520. [Val Acc 64%, Loss 0.801580]
Iter 530. [Val Acc 64%, Loss 0.800029]
Iter 540. [Val Acc 64%, Loss 0.798580]
Iter 550. [Val Acc 64%, Loss 0.797229]
Iter 560. [Val Acc 64%, Loss 0.795180]
Iter 570. [Val Acc 64%, Loss 0.793616]
Iter 580. [Val Acc 64%, Loss 0.792142]
Iter 590. [Val Acc 64%, Loss 0.790722]
Iter 600. [Val Acc 64%, Loss 0.789357]
Iter 610. [Val Acc 64%, Loss 0.787887]
Iter 620. [Val Acc 64%, Loss 0.786438]
Iter 630. [Val Acc 65%, Loss 0.784737]
Iter 640. [Val Acc 65%, Loss 0.783578]
Iter 650. [Val Acc 65%, Loss 0.782101]
Iter 660. [Val Acc 65%, Loss 0.780594]
Iter 670. [Val Acc 65%, Loss 0.779144]
Iter 680. [Val Acc 65%, Loss 0.777746]
Iter 690. [Val Acc 65%, Loss 0.776177]
Iter 700. [Val Acc 65%, Loss 0.774936]
Iter 710. [Val Acc 65%, Loss 0.773153]
Iter 720. [Val Acc 65%, Loss 0.771496]
Iter 730. [Val Acc 65%, Loss 0.770301]
Iter 740. [Val Acc 65%, Loss 0.768918]
Iter 750. [Val Acc 65%, Loss 0.767392]
Iter 760. [Val Acc 65%, Loss 0.766347]
Iter 770. [Val Acc 65%, Loss 0.765385]
Iter 780. [Val Acc 65%, Loss 0.764265]
Iter 790. [Val Acc 65%, Loss 0.762733]
Iter 800. [Val Acc 65%, Loss 0.761390]
Iter 810. [Val Acc 65%, Loss 0.760108]
Iter 820. [Val Acc 65%, Loss 0.758848]
Iter 830. [Val Acc 65%, Loss 0.757540]
Iter 840. [Val Acc 65%, Loss 0.756335]
Iter 850. [Val Acc 65%, Loss 0.754976]
Iter 860. [Val Acc 65%, Loss 0.754003]
Iter 870. [Val Acc 65%, Loss 0.752844]
Iter 880. [Val Acc 65%, Loss 0.751843]
Iter 890. [Val Acc 65%, Loss 0.750819]
Iter 900. [Val Acc 65%, Loss 0.749556]
Iter 910. [Val Acc 65%, Loss 0.748219]
Iter 920. [Val Acc 65%, Loss 0.747076]
Iter 930. [Val Acc 66%, Loss 0.745959]
Iter 940. [Val Acc 66%, Loss 0.744811]
Iter 950. [Val Acc 66%, Loss 0.743666]
Iter 960. [Val Acc 66%, Loss 0.742686]
Iter 970. [Val Acc 66%, Loss 0.741863]
Iter 980. [Val Acc 66%, Loss 0.740832]
Iter 990. [Val Acc 66%, Loss 0.739786]
Iter 1000. [Val Acc 66%, Loss 0.738588]
-------------------------
mu= 0.01 batch_size= 50000 :
Iter 10. [Val Acc 65%, Loss 0.755230]
Iter 20. [Val Acc 65%, Loss 0.753799]
Iter 30. [Val Acc 65%, Loss 0.752042]
Iter 40. [Val Acc 66%, Loss 0.750682]
Iter 50. [Val Acc 66%, Loss 0.748914]
Iter 60. [Val Acc 66%, Loss 0.747344]
Iter 70. [Val Acc 66%, Loss 0.745985]
Iter 80. [Val Acc 66%, Loss 0.744480]
Iter 90. [Val Acc 66%, Loss 0.743040]
Iter 100. [Val Acc 66%, Loss 0.741452]
Iter 110. [Val Acc 66%, Loss 0.740027]
Iter 120. [Val Acc 66%, Loss 0.738746]
Iter 130. [Val Acc 66%, Loss 0.737431]
Iter 140. [Val Acc 66%, Loss 0.736303]
Iter 150. [Val Acc 66%, Loss 0.734928]
Iter 160. [Val Acc 66%, Loss 0.733650]
Iter 170. [Val Acc 66%, Loss 0.732754]
Iter 180. [Val Acc 66%, Loss 0.731638]
Iter 190. [Val Acc 66%, Loss 0.730632]
Iter 200. [Val Acc 66%, Loss 0.729331]
Iter 210. [Val Acc 66%, Loss 0.728069]
Iter 220. [Val Acc 66%, Loss 0.726973]
Iter 230. [Val Acc 66%, Loss 0.725977]
Iter 240. [Val Acc 66%, Loss 0.724940]
Iter 250. [Val Acc 67%, Loss 0.723628]
Iter 260. [Val Acc 67%, Loss 0.722392]
Iter 270. [Val Acc 67%, Loss 0.721598]
Iter 280. [Val Acc 67%, Loss 0.720821]
Iter 290. [Val Acc 67%, Loss 0.719714]
Iter 300. [Val Acc 67%, Loss 0.718522]
Iter 310. [Val Acc 67%, Loss 0.717457]
Iter 320. [Val Acc 67%, Loss 0.716587]
Iter 330. [Val Acc 67%, Loss 0.715639]
Iter 340. [Val Acc 67%, Loss 0.714676]
Iter 350. [Val Acc 67%, Loss 0.713573]
Iter 360. [Val Acc 67%, Loss 0.712725]
Iter 370. [Val Acc 67%, Loss 0.711915]
Iter 380. [Val Acc 67%, Loss 0.710702]
Iter 390. [Val Acc 67%, Loss 0.709541]
Iter 400. [Val Acc 67%, Loss 0.708734]
Iter 410. [Val Acc 67%, Loss 0.707741]
Iter 420. [Val Acc 67%, Loss 0.706738]
Iter 430. [Val Acc 67%, Loss 0.705661]
Iter 440. [Val Acc 67%, Loss 0.704696]
Iter 450. [Val Acc 67%, Loss 0.703845]
Iter 460. [Val Acc 67%, Loss 0.703125]
Iter 470. [Val Acc 67%, Loss 0.702134]
Iter 480. [Val Acc 67%, Loss 0.701258]
Iter 490. [Val Acc 67%, Loss 0.700424]
Iter 500. [Val Acc 67%, Loss 0.699536]
Iter 510. [Val Acc 67%, Loss 0.698573]
Iter 520. [Val Acc 67%, Loss 0.697681]
Iter 530. [Val Acc 67%, Loss 0.696743]
Iter 540. [Val Acc 67%, Loss 0.695802]
Iter 550. [Val Acc 67%, Loss 0.694991]
Iter 560. [Val Acc 67%, Loss 0.694154]
Iter 570. [Val Acc 67%, Loss 0.693397]
Iter 580. [Val Acc 67%, Loss 0.692732]
Iter 590. [Val Acc 67%, Loss 0.691846]
Iter 600. [Val Acc 67%, Loss 0.691009]
Iter 610. [Val Acc 67%, Loss 0.690373]
Iter 620. [Val Acc 67%, Loss 0.689547]
Iter 630. [Val Acc 67%, Loss 0.688896]
Iter 640. [Val Acc 68%, Loss 0.688155]
Iter 650. [Val Acc 68%, Loss 0.687193]
Iter 660. [Val Acc 68%, Loss 0.686396]
Iter 670. [Val Acc 68%, Loss 0.685603]
Iter 680. [Val Acc 68%, Loss 0.684812]
Iter 690. [Val Acc 68%, Loss 0.684158]
Iter 700. [Val Acc 68%, Loss 0.683343]
Iter 710. [Val Acc 68%, Loss 0.682598]
Iter 720. [Val Acc 68%, Loss 0.682021]
Iter 730. [Val Acc 68%, Loss 0.681436]
Iter 740. [Val Acc 68%, Loss 0.680796]
Iter 750. [Val Acc 68%, Loss 0.679902]
Iter 760. [Val Acc 68%, Loss 0.679117]
Iter 770. [Val Acc 68%, Loss 0.678494]
Iter 780. [Val Acc 68%, Loss 0.677840]
Iter 790. [Val Acc 68%, Loss 0.677069]
Iter 800. [Val Acc 68%, Loss 0.676521]
Iter 810. [Val Acc 68%, Loss 0.675886]
Iter 820. [Val Acc 68%, Loss 0.675161]
Iter 830. [Val Acc 68%, Loss 0.674611]
Iter 840. [Val Acc 68%, Loss 0.674039]
Iter 850. [Val Acc 68%, Loss 0.673409]
Iter 860. [Val Acc 68%, Loss 0.672753]
Iter 870. [Val Acc 68%, Loss 0.672043]
Iter 880. [Val Acc 68%, Loss 0.671548]
Iter 890. [Val Acc 68%, Loss 0.671124]
Iter 900. [Val Acc 68%, Loss 0.670509]
Iter 910. [Val Acc 68%, Loss 0.669867]
Iter 920. [Val Acc 68%, Loss 0.669327]
Iter 930. [Val Acc 68%, Loss 0.668823]
Iter 940. [Val Acc 68%, Loss 0.668254]
Iter 950. [Val Acc 68%, Loss 0.667773]
Iter 960. [Val Acc 68%, Loss 0.667096]
Iter 970. [Val Acc 68%, Loss 0.666527]
Iter 980. [Val Acc 68%, Loss 0.666055]
Iter 990. [Val Acc 68%, Loss 0.665470]
Iter 1000. [Val Acc 68%, Loss 0.664856]
-------------------------
```

```
mu= 0.1 batch_size= 1 :
Iter 10. [Val Acc 68%, Loss 0.679409]
Iter 20. [Val Acc 68%, Loss 0.671601]
Iter 30. [Val Acc 68%, Loss 0.665270]
Iter 40. [Val Acc 69%, Loss 0.660963]
Iter 50. [Val Acc 69%, Loss 0.652989]
Iter 60. [Val Acc 69%, Loss 0.647111]
Iter 70. [Val Acc 69%, Loss 0.645079]
Iter 80. [Val Acc 70%, Loss 0.639254]
Iter 90. [Val Acc 70%, Loss 0.635633]
Iter 100. [Val Acc 70%, Loss 0.632096]
Iter 110. [Val Acc 70%, Loss 0.628129]
Iter 120. [Val Acc 70%, Loss 0.625318]
Iter 130. [Val Acc 70%, Loss 0.623155]
Iter 140. [Val Acc 71%, Loss 0.621235]
Iter 150. [Val Acc 71%, Loss 0.616948]
Iter 160. [Val Acc 71%, Loss 0.615482]
Iter 170. [Val Acc 71%, Loss 0.612431]
Iter 180. [Val Acc 70%, Loss 0.612584]
Iter 190. [Val Acc 71%, Loss 0.610688]
Iter 200. [Val Acc 71%, Loss 0.608137]
Iter 210. [Val Acc 71%, Loss 0.603869]
Iter 220. [Val Acc 71%, Loss 0.603650]
Iter 230. [Val Acc 71%, Loss 0.602367]
Iter 240. [Val Acc 71%, Loss 0.599944]
Iter 250. [Val Acc 71%, Loss 0.597481]
Iter 260. [Val Acc 71%, Loss 0.596554]
Iter 270. [Val Acc 71%, Loss 0.596659]
Iter 280. [Val Acc 71%, Loss 0.595849]
Iter 290. [Val Acc 71%, Loss 0.597362]
Iter 300. [Val Acc 72%, Loss 0.592679]
Iter 310. [Val Acc 72%, Loss 0.591685]
Iter 320. [Val Acc 72%, Loss 0.592053]
Iter 330. [Val Acc 72%, Loss 0.591132]
Iter 340. [Val Acc 72%, Loss 0.589139]
Iter 350. [Val Acc 72%, Loss 0.586347]
Iter 360. [Val Acc 72%, Loss 0.584569]
Iter 370. [Val Acc 72%, Loss 0.586159]
Iter 380. [Val Acc 72%, Loss 0.585876]
Iter 390. [Val Acc 72%, Loss 0.584457]
Iter 400. [Val Acc 72%, Loss 0.583038]
Iter 410. [Val Acc 72%, Loss 0.583724]
Iter 420. [Val Acc 72%, Loss 0.583272]
Iter 430. [Val Acc 72%, Loss 0.580109]
Iter 440. [Val Acc 72%, Loss 0.580403]
Iter 450. [Val Acc 72%, Loss 0.580265]
Iter 460. [Val Acc 72%, Loss 0.581923]
Iter 470. [Val Acc 73%, Loss 0.578253]
Iter 480. [Val Acc 73%, Loss 0.579597]
Iter 490. [Val Acc 73%, Loss 0.577945]
Iter 500. [Val Acc 73%, Loss 0.576872]
Iter 510. [Val Acc 72%, Loss 0.580532]
Iter 520. [Val Acc 73%, Loss 0.577793]
Iter 530. [Val Acc 73%, Loss 0.575839]
Iter 540. [Val Acc 73%, Loss 0.573407]
Iter 550. [Val Acc 73%, Loss 0.572914]
Iter 560. [Val Acc 73%, Loss 0.571298]
Iter 570. [Val Acc 73%, Loss 0.573441]
Iter 580. [Val Acc 73%, Loss 0.573641]
Iter 590. [Val Acc 73%, Loss 0.569347]
Iter 600. [Val Acc 73%, Loss 0.570235]
Iter 610. [Val Acc 73%, Loss 0.570195]
Iter 620. [Val Acc 73%, Loss 0.570531]
Iter 630. [Val Acc 73%, Loss 0.569130]
Iter 640. [Val Acc 73%, Loss 0.570484]
Iter 650. [Val Acc 73%, Loss 0.570825]
Iter 660. [Val Acc 73%, Loss 0.570190]
Iter 670. [Val Acc 73%, Loss 0.568838]
Iter 680. [Val Acc 73%, Loss 0.568787]
Iter 690. [Val Acc 73%, Loss 0.573686]
Iter 700. [Val Acc 73%, Loss 0.569949]
Iter 710. [Val Acc 73%, Loss 0.569296]
Iter 720. [Val Acc 73%, Loss 0.567528]
Iter 730. [Val Acc 73%, Loss 0.567788]
Iter 740. [Val Acc 73%, Loss 0.565928]
Iter 750. [Val Acc 73%, Loss 0.567437]
Iter 760. [Val Acc 73%, Loss 0.566625]
Iter 770. [Val Acc 73%, Loss 0.565927]
Iter 780. [Val Acc 73%, Loss 0.565327]
Iter 790. [Val Acc 73%, Loss 0.564760]
Iter 800. [Val Acc 73%, Loss 0.565287]
Iter 810. [Val Acc 73%, Loss 0.564235]
Iter 820. [Val Acc 73%, Loss 0.564770]
Iter 830. [Val Acc 73%, Loss 0.565330]
Iter 840. [Val Acc 73%, Loss 0.570959]
Iter 850. [Val Acc 73%, Loss 0.567317]
Iter 860. [Val Acc 73%, Loss 0.566348]
Iter 870. [Val Acc 73%, Loss 0.570058]
Iter 880. [Val Acc 73%, Loss 0.565754]
Iter 890. [Val Acc 73%, Loss 0.567168]
Iter 900. [Val Acc 73%, Loss 0.567377]
Iter 910. [Val Acc 73%, Loss 0.570124]
Iter 920. [Val Acc 73%, Loss 0.569124]
Iter 930. [Val Acc 73%, Loss 0.563923]
Iter 940. [Val Acc 73%, Loss 0.567770]
Iter 950. [Val Acc 73%, Loss 0.565879]
Iter 960. [Val Acc 73%, Loss 0.564209]
Iter 970. [Val Acc 73%, Loss 0.565112]
Iter 980. [Val Acc 73%, Loss 0.563434]
Iter 990. [Val Acc 73%, Loss 0.563550]
Iter 1000. [Val Acc 73%, Loss 0.563739]
-------------------------
mu= 0.1 batch_size= 64 :
Iter 10. [Val Acc 70%, Loss 0.592290]
Iter 20. [Val Acc 70%, Loss 0.585235]
Iter 30. [Val Acc 71%, Loss 0.579702]
Iter 40. [Val Acc 72%, Loss 0.572464]
Iter 50. [Val Acc 72%, Loss 0.571763]
Iter 60. [Val Acc 73%, Loss 0.571309]
Iter 70. [Val Acc 73%, Loss 0.571737]
Iter 80. [Val Acc 73%, Loss 0.567088]
Iter 90. [Val Acc 73%, Loss 0.565484]
Iter 100. [Val Acc 73%, Loss 0.565041]
Iter 110. [Val Acc 73%, Loss 0.563843]
Iter 120. [Val Acc 73%, Loss 0.567036]
Iter 130. [Val Acc 73%, Loss 0.563507]
Iter 140. [Val Acc 73%, Loss 0.563263]
Iter 150. [Val Acc 73%, Loss 0.562121]
Iter 160. [Val Acc 73%, Loss 0.562797]
Iter 170. [Val Acc 73%, Loss 0.564114]
Iter 180. [Val Acc 73%, Loss 0.563105]
Iter 190. [Val Acc 73%, Loss 0.562744]
Iter 200. [Val Acc 73%, Loss 0.562479]
Iter 210. [Val Acc 73%, Loss 0.562616]
Iter 220. [Val Acc 73%, Loss 0.566107]
Iter 230. [Val Acc 73%, Loss 0.564263]
Iter 240. [Val Acc 73%, Loss 0.562913]
Iter 250. [Val Acc 73%, Loss 0.562721]
Iter 260. [Val Acc 73%, Loss 0.561437]
Iter 270. [Val Acc 73%, Loss 0.562036]
Iter 280. [Val Acc 73%, Loss 0.561121]
Iter 290. [Val Acc 73%, Loss 0.562473]
Iter 300. [Val Acc 73%, Loss 0.562500]
Iter 310. [Val Acc 73%, Loss 0.563313]
Iter 320. [Val Acc 73%, Loss 0.562950]
Iter 330. [Val Acc 73%, Loss 0.561976]
Iter 340. [Val Acc 73%, Loss 0.561059]
Iter 350. [Val Acc 73%, Loss 0.560430]
Iter 360. [Val Acc 73%, Loss 0.561136]
Iter 370. [Val Acc 73%, Loss 0.560419]
Iter 380. [Val Acc 73%, Loss 0.560722]
Iter 390. [Val Acc 73%, Loss 0.560222]
Iter 400. [Val Acc 73%, Loss 0.561431]
Iter 410. [Val Acc 73%, Loss 0.560787]
Iter 420. [Val Acc 73%, Loss 0.560813]
Iter 430. [Val Acc 73%, Loss 0.563041]
Iter 440. [Val Acc 73%, Loss 0.562902]
Iter 450. [Val Acc 73%, Loss 0.562116]
Iter 460. [Val Acc 73%, Loss 0.562575]
Iter 470. [Val Acc 73%, Loss 0.562176]
Iter 480. [Val Acc 73%, Loss 0.561807]
Iter 490. [Val Acc 73%, Loss 0.561357]
Iter 500. [Val Acc 73%, Loss 0.561262]
Iter 510. [Val Acc 73%, Loss 0.564092]
Iter 520. [Val Acc 73%, Loss 0.561313]
Iter 530. [Val Acc 73%, Loss 0.560520]
Iter 540. [Val Acc 73%, Loss 0.561145]
Iter 550. [Val Acc 73%, Loss 0.563100]
Iter 560. [Val Acc 73%, Loss 0.562265]
Iter 570. [Val Acc 73%, Loss 0.562030]
Iter 580. [Val Acc 73%, Loss 0.559475]
Iter 590. [Val Acc 73%, Loss 0.560141]
Iter 600. [Val Acc 73%, Loss 0.561045]
Iter 610. [Val Acc 73%, Loss 0.561885]
Iter 620. [Val Acc 73%, Loss 0.560077]
Iter 630. [Val Acc 74%, Loss 0.559479]
Iter 640. [Val Acc 74%, Loss 0.559959]
Iter 650. [Val Acc 73%, Loss 0.560518]
Iter 660. [Val Acc 73%, Loss 0.561056]
Iter 670. [Val Acc 73%, Loss 0.559532]
Iter 680. [Val Acc 74%, Loss 0.559924]
Iter 690. [Val Acc 73%, Loss 0.561509]
Iter 700. [Val Acc 73%, Loss 0.560791]
Iter 710. [Val Acc 73%, Loss 0.560777]
Iter 720. [Val Acc 74%, Loss 0.559533]
Iter 730. [Val Acc 74%, Loss 0.559723]
Iter 740. [Val Acc 74%, Loss 0.560264]
Iter 750. [Val Acc 74%, Loss 0.560015]
Iter 760. [Val Acc 73%, Loss 0.562035]
Iter 770. [Val Acc 73%, Loss 0.561162]
Iter 780. [Val Acc 73%, Loss 0.561993]
Iter 790. [Val Acc 73%, Loss 0.562308]
Iter 800. [Val Acc 74%, Loss 0.560469]
Iter 810. [Val Acc 73%, Loss 0.560603]
Iter 820. [Val Acc 73%, Loss 0.562177]
Iter 830. [Val Acc 73%, Loss 0.559160]
Iter 840. [Val Acc 73%, Loss 0.560149]
Iter 850. [Val Acc 74%, Loss 0.560336]
Iter 860. [Val Acc 73%, Loss 0.560631]
Iter 870. [Val Acc 73%, Loss 0.560824]
Iter 880. [Val Acc 73%, Loss 0.561900]
Iter 890. [Val Acc 74%, Loss 0.560320]
Iter 900. [Val Acc 73%, Loss 0.561141]
Iter 910. [Val Acc 73%, Loss 0.561781]
Iter 920. [Val Acc 73%, Loss 0.562192]
Iter 930. [Val Acc 73%, Loss 0.561363]
Iter 940. [Val Acc 74%, Loss 0.560408]
Iter 950. [Val Acc 73%, Loss 0.560634]
Iter 960. [Val Acc 73%, Loss 0.561237]
Iter 970. [Val Acc 73%, Loss 0.560777]
Iter 980. [Val Acc 73%, Loss 0.560328]
Iter 990. [Val Acc 73%, Loss 0.561638]
Iter 1000. [Val Acc 73%, Loss 0.561736]
-------------------------
mu= 0.1 batch_size= 1024 :
Iter 10. [Val Acc 70%, Loss 0.588655]
Iter 20. [Val Acc 71%, Loss 0.579326]
Iter 30. [Val Acc 72%, Loss 0.572884]
Iter 40. [Val Acc 72%, Loss 0.568879]
Iter 50. [Val Acc 73%, Loss 0.566292]
Iter 60. [Val Acc 73%, Loss 0.564891]
Iter 70. [Val Acc 73%, Loss 0.561464]
Iter 80. [Val Acc 73%, Loss 0.561306]
Iter 90. [Val Acc 73%, Loss 0.561099]
Iter 100. [Val Acc 73%, Loss 0.562469]
Iter 110. [Val Acc 73%, Loss 0.563372]
Iter 120. [Val Acc 73%, Loss 0.561993]
Iter 130. [Val Acc 73%, Loss 0.561547]
Iter 140. [Val Acc 73%, Loss 0.561684]
Iter 150. [Val Acc 73%, Loss 0.558833]
Iter 160. [Val Acc 73%, Loss 0.559328]
Iter 170. [Val Acc 73%, Loss 0.559064]
Iter 180. [Val Acc 73%, Loss 0.562386]
Iter 190. [Val Acc 73%, Loss 0.562403]
Iter 200. [Val Acc 73%, Loss 0.561534]
Iter 210. [Val Acc 73%, Loss 0.561327]
Iter 220. [Val Acc 73%, Loss 0.560810]
Iter 230. [Val Acc 74%, Loss 0.560575]
Iter 240. [Val Acc 74%, Loss 0.559611]
Iter 250. [Val Acc 74%, Loss 0.559275]
Iter 260. [Val Acc 73%, Loss 0.559620]
Iter 270. [Val Acc 73%, Loss 0.562252]
Iter 280. [Val Acc 73%, Loss 0.560616]
Iter 290. [Val Acc 74%, Loss 0.562359]
Iter 300. [Val Acc 73%, Loss 0.560249]
Iter 310. [Val Acc 73%, Loss 0.561329]
Iter 320. [Val Acc 73%, Loss 0.564209]
Iter 330. [Val Acc 73%, Loss 0.561896]
Iter 340. [Val Acc 73%, Loss 0.560753]
Iter 350. [Val Acc 73%, Loss 0.560915]
Iter 360. [Val Acc 73%, Loss 0.560551]
Iter 370. [Val Acc 73%, Loss 0.560028]
Iter 380. [Val Acc 73%, Loss 0.559377]
Iter 390. [Val Acc 74%, Loss 0.559433]
Iter 400. [Val Acc 73%, Loss 0.560036]
Iter 410. [Val Acc 73%, Loss 0.559280]
Iter 420. [Val Acc 73%, Loss 0.559554]
Iter 430. [Val Acc 73%, Loss 0.559622]
Iter 440. [Val Acc 73%, Loss 0.562370]
Iter 450. [Val Acc 73%, Loss 0.560087]
Iter 460. [Val Acc 73%, Loss 0.561199]
Iter 470. [Val Acc 73%, Loss 0.560547]
Iter 480. [Val Acc 73%, Loss 0.560935]
Iter 490. [Val Acc 73%, Loss 0.560348]
Iter 500. [Val Acc 73%, Loss 0.560621]
Iter 510. [Val Acc 73%, Loss 0.560382]
Iter 520. [Val Acc 73%, Loss 0.561751]
Iter 530. [Val Acc 73%, Loss 0.563205]
Iter 540. [Val Acc 73%, Loss 0.564479]
Iter 550. [Val Acc 73%, Loss 0.565028]
Iter 560. [Val Acc 73%, Loss 0.561909]
Iter 570. [Val Acc 73%, Loss 0.561870]
Iter 580. [Val Acc 73%, Loss 0.563272]
Iter 590. [Val Acc 73%, Loss 0.561354]
Iter 600. [Val Acc 73%, Loss 0.561165]
Iter 610. [Val Acc 73%, Loss 0.559553]
Iter 620. [Val Acc 73%, Loss 0.562268]
Iter 630. [Val Acc 73%, Loss 0.559616]
Iter 640. [Val Acc 73%, Loss 0.560678]
Iter 650. [Val Acc 73%, Loss 0.560354]
Iter 660. [Val Acc 73%, Loss 0.563479]
Iter 670. [Val Acc 74%, Loss 0.560747]
Iter 680. [Val Acc 73%, Loss 0.559797]
Iter 690. [Val Acc 73%, Loss 0.561270]
Iter 700. [Val Acc 74%, Loss 0.559975]
Iter 710. [Val Acc 73%, Loss 0.565731]
Iter 720. [Val Acc 74%, Loss 0.562695]
Iter 730. [Val Acc 73%, Loss 0.560558]
Iter 740. [Val Acc 73%, Loss 0.561093]
Iter 750. [Val Acc 74%, Loss 0.561564]
Iter 760. [Val Acc 74%, Loss 0.560283]
Iter 770. [Val Acc 73%, Loss 0.560645]
Iter 780. [Val Acc 73%, Loss 0.561482]
Iter 790. [Val Acc 74%, Loss 0.559948]
Iter 800. [Val Acc 73%, Loss 0.560110]
Iter 810. [Val Acc 73%, Loss 0.561499]
Iter 820. [Val Acc 73%, Loss 0.560626]
Iter 830. [Val Acc 73%, Loss 0.561546]
Iter 840. [Val Acc 73%, Loss 0.562858]
Iter 850. [Val Acc 73%, Loss 0.560825]
Iter 860. [Val Acc 73%, Loss 0.561020]
Iter 870. [Val Acc 73%, Loss 0.560428]
Iter 880. [Val Acc 73%, Loss 0.558847]
Iter 890. [Val Acc 73%, Loss 0.560968]
Iter 900. [Val Acc 73%, Loss 0.560754]
Iter 910. [Val Acc 73%, Loss 0.559668]
Iter 920. [Val Acc 73%, Loss 0.559342]
Iter 930. [Val Acc 73%, Loss 0.560140]
Iter 940. [Val Acc 73%, Loss 0.558062]
Iter 950. [Val Acc 73%, Loss 0.559619]
Iter 960. [Val Acc 73%, Loss 0.561443]
Iter 970. [Val Acc 73%, Loss 0.559474]
Iter 980. [Val Acc 73%, Loss 0.560724]
Iter 990. [Val Acc 73%, Loss 0.563589]
Iter 1000. [Val Acc 73%, Loss 0.559996]
-------------------------
mu= 0.1 batch_size= 50000 :
Iter 10. [Val Acc 70%, Loss 0.587093]
Iter 20. [Val Acc 71%, Loss 0.581235]
Iter 30. [Val Acc 72%, Loss 0.574132]
Iter 40. [Val Acc 72%, Loss 0.571098]
Iter 50. [Val Acc 72%, Loss 0.568353]
Iter 60. [Val Acc 73%, Loss 0.564279]
Iter 70. [Val Acc 73%, Loss 0.564496]
Iter 80. [Val Acc 73%, Loss 0.562809]
Iter 90. [Val Acc 73%, Loss 0.561855]
Iter 100. [Val Acc 73%, Loss 0.560910]
Iter 110. [Val Acc 73%, Loss 0.560603]
Iter 120. [Val Acc 73%, Loss 0.561356]
Iter 130. [Val Acc 73%, Loss 0.562828]
Iter 140. [Val Acc 73%, Loss 0.561949]
Iter 150. [Val Acc 73%, Loss 0.560387]
Iter 160. [Val Acc 73%, Loss 0.561294]
Iter 170. [Val Acc 73%, Loss 0.560468]
Iter 180. [Val Acc 73%, Loss 0.564345]
Iter 190. [Val Acc 73%, Loss 0.559498]
Iter 200. [Val Acc 74%, Loss 0.562014]
Iter 210. [Val Acc 73%, Loss 0.559234]
Iter 220. [Val Acc 73%, Loss 0.559281]
Iter 230. [Val Acc 73%, Loss 0.559418]
Iter 240. [Val Acc 73%, Loss 0.561295]
Iter 250. [Val Acc 73%, Loss 0.559458]
Iter 260. [Val Acc 74%, Loss 0.559566]
Iter 270. [Val Acc 73%, Loss 0.560627]
Iter 280. [Val Acc 73%, Loss 0.559881]
Iter 290. [Val Acc 73%, Loss 0.558768]
Iter 300. [Val Acc 73%, Loss 0.559939]
Iter 310. [Val Acc 73%, Loss 0.560570]
Iter 320. [Val Acc 73%, Loss 0.561866]
Iter 330. [Val Acc 73%, Loss 0.559524]
Iter 340. [Val Acc 73%, Loss 0.559664]
Iter 350. [Val Acc 73%, Loss 0.559450]
Iter 360. [Val Acc 73%, Loss 0.559615]
Iter 370. [Val Acc 73%, Loss 0.559145]
Iter 380. [Val Acc 74%, Loss 0.559157]
Iter 390. [Val Acc 73%, Loss 0.559673]
Iter 400. [Val Acc 74%, Loss 0.559804]
Iter 410. [Val Acc 73%, Loss 0.562760]
Iter 420. [Val Acc 73%, Loss 0.560370]
Iter 430. [Val Acc 73%, Loss 0.561171]
Iter 440. [Val Acc 73%, Loss 0.564657]
Iter 450. [Val Acc 73%, Loss 0.561261]
Iter 460. [Val Acc 73%, Loss 0.563374]
Iter 470. [Val Acc 73%, Loss 0.559754]
Iter 480. [Val Acc 73%, Loss 0.560187]
Iter 490. [Val Acc 73%, Loss 0.559167]
Iter 500. [Val Acc 73%, Loss 0.561825]
Iter 510. [Val Acc 73%, Loss 0.558672]
Iter 520. [Val Acc 73%, Loss 0.560738]
Iter 530. [Val Acc 73%, Loss 0.560045]
Iter 540. [Val Acc 73%, Loss 0.559471]
Iter 550. [Val Acc 73%, Loss 0.560768]
Iter 560. [Val Acc 73%, Loss 0.560978]
Iter 570. [Val Acc 74%, Loss 0.559859]
Iter 580. [Val Acc 73%, Loss 0.560666]
Iter 590. [Val Acc 74%, Loss 0.559373]
Iter 600. [Val Acc 74%, Loss 0.558546]
Iter 610. [Val Acc 74%, Loss 0.558270]
Iter 620. [Val Acc 74%, Loss 0.559078]
Iter 630. [Val Acc 73%, Loss 0.559734]
Iter 640. [Val Acc 74%, Loss 0.559508]
Iter 650. [Val Acc 73%, Loss 0.564200]
Iter 660. [Val Acc 73%, Loss 0.560632]
Iter 670. [Val Acc 73%, Loss 0.559314]
Iter 680. [Val Acc 73%, Loss 0.561126]
Iter 690. [Val Acc 74%, Loss 0.559224]
Iter 700. [Val Acc 73%, Loss 0.560274]
Iter 710. [Val Acc 73%, Loss 0.560439]
Iter 720. [Val Acc 74%, Loss 0.558450]
Iter 730. [Val Acc 74%, Loss 0.558533]
Iter 740. [Val Acc 74%, Loss 0.558783]
Iter 750. [Val Acc 73%, Loss 0.560155]
Iter 760. [Val Acc 74%, Loss 0.558911]
Iter 770. [Val Acc 73%, Loss 0.564054]
Iter 780. [Val Acc 73%, Loss 0.560115]
Iter 790. [Val Acc 73%, Loss 0.559701]
Iter 800. [Val Acc 73%, Loss 0.560570]
Iter 810. [Val Acc 73%, Loss 0.559442]
Iter 820. [Val Acc 73%, Loss 0.560083]
Iter 830. [Val Acc 73%, Loss 0.560481]
Iter 840. [Val Acc 73%, Loss 0.559980]
Iter 850. [Val Acc 73%, Loss 0.559379]
Iter 860. [Val Acc 73%, Loss 0.559846]
Iter 870. [Val Acc 73%, Loss 0.561364]
Iter 880. [Val Acc 73%, Loss 0.562525]
Iter 890. [Val Acc 73%, Loss 0.561186]
Iter 900. [Val Acc 73%, Loss 0.559891]
Iter 910. [Val Acc 73%, Loss 0.560325]
Iter 920. [Val Acc 73%, Loss 0.561209]
Iter 930. [Val Acc 73%, Loss 0.559837]
Iter 940. [Val Acc 73%, Loss 0.558987]
Iter 950. [Val Acc 73%, Loss 0.561734]
Iter 960. [Val Acc 73%, Loss 0.559366]
Iter 970. [Val Acc 73%, Loss 0.560490]
Iter 980. [Val Acc 73%, Loss 0.559810]
Iter 990. [Val Acc 74%, Loss 0.558847]
Iter 1000. [Val Acc 74%, Loss 0.558762]
-------------------------
mu= 0.5 batch_size= 1 :
Iter 10. [Val Acc 71%, Loss 0.581826]
Iter 20. [Val Acc 71%, Loss 0.587028]
Iter 30. [Val Acc 72%, Loss 0.583999]
Iter 40. [Val Acc 73%, Loss 0.574338]
Iter 50. [Val Acc 72%, Loss 0.581830]
Iter 60. [Val Acc 71%, Loss 0.592591]
Iter 70. [Val Acc 71%, Loss 0.584839]
Iter 80. [Val Acc 72%, Loss 0.582638]
Iter 90. [Val Acc 72%, Loss 0.580157]
Iter 100. [Val Acc 72%, Loss 0.582450]
Iter 110. [Val Acc 72%, Loss 0.582991]
Iter 120. [Val Acc 71%, Loss 0.584508]
Iter 130. [Val Acc 71%, Loss 0.580782]
Iter 140. [Val Acc 73%, Loss 0.578765]
Iter 150. [Val Acc 72%, Loss 0.577924]
Iter 160. [Val Acc 72%, Loss 0.589741]
Iter 170. [Val Acc 71%, Loss 0.595559]
Iter 180. [Val Acc 71%, Loss 0.589188]
Iter 190. [Val Acc 72%, Loss 0.581944]
Iter 200. [Val Acc 70%, Loss 0.629927]
Iter 210. [Val Acc 72%, Loss 0.579404]
Iter 220. [Val Acc 71%, Loss 0.592928]
Iter 230. [Val Acc 72%, Loss 0.583686]
Iter 240. [Val Acc 71%, Loss 0.603126]
Iter 250. [Val Acc 72%, Loss 0.580994]
Iter 260. [Val Acc 72%, Loss 0.588131]
Iter 270. [Val Acc 72%, Loss 0.586901]
Iter 280. [Val Acc 72%, Loss 0.584389]
Iter 290. [Val Acc 73%, Loss 0.583983]
Iter 300. [Val Acc 71%, Loss 0.592173]
Iter 310. [Val Acc 72%, Loss 0.581581]
Iter 320. [Val Acc 72%, Loss 0.589940]
Iter 330. [Val Acc 72%, Loss 0.585270]
Iter 340. [Val Acc 71%, Loss 0.599005]
Iter 350. [Val Acc 72%, Loss 0.584067]
Iter 360. [Val Acc 72%, Loss 0.575166]
Iter 370. [Val Acc 72%, Loss 0.587395]
Iter 380. [Val Acc 71%, Loss 0.588687]
Iter 390. [Val Acc 72%, Loss 0.588800]
Iter 400. [Val Acc 72%, Loss 0.586720]
Iter 410. [Val Acc 72%, Loss 0.582128]
Iter 420. [Val Acc 71%, Loss 0.598927]
Iter 430. [Val Acc 71%, Loss 0.602086]
Iter 440. [Val Acc 73%, Loss 0.572028]
Iter 450. [Val Acc 72%, Loss 0.581195]
Iter 460. [Val Acc 72%, Loss 0.584306]
Iter 470. [Val Acc 72%, Loss 0.575389]
Iter 480. [Val Acc 71%, Loss 0.592165]
Iter 490. [Val Acc 70%, Loss 0.604073]
Iter 500. [Val Acc 72%, Loss 0.587972]
Iter 510. [Val Acc 70%, Loss 0.607682]
Iter 520. [Val Acc 72%, Loss 0.580047]
Iter 530. [Val Acc 72%, Loss 0.578526]
Iter 540. [Val Acc 72%, Loss 0.588844]
Iter 550. [Val Acc 73%, Loss 0.579851]
Iter 560. [Val Acc 72%, Loss 0.589152]
Iter 570. [Val Acc 72%, Loss 0.585632]
Iter 580. [Val Acc 72%, Loss 0.580526]
Iter 590. [Val Acc 72%, Loss 0.580519]
Iter 600. [Val Acc 72%, Loss 0.576636]
Iter 610. [Val Acc 72%, Loss 0.574260]
Iter 620. [Val Acc 73%, Loss 0.573735]
Iter 630. [Val Acc 72%, Loss 0.577194]
Iter 640. [Val Acc 73%, Loss 0.578603]
Iter 650. [Val Acc 72%, Loss 0.595883]
Iter 660. [Val Acc 72%, Loss 0.582291]
Iter 670. [Val Acc 73%, Loss 0.578250]
Iter 680. [Val Acc 70%, Loss 0.602881]
Iter 690. [Val Acc 72%, Loss 0.584581]
Iter 700. [Val Acc 72%, Loss 0.577787]
Iter 710. [Val Acc 72%, Loss 0.580681]
Iter 720. [Val Acc 71%, Loss 0.597347]
Iter 730. [Val Acc 72%, Loss 0.578597]
Iter 740. [Val Acc 72%, Loss 0.578396]
Iter 750. [Val Acc 72%, Loss 0.590653]
Iter 760. [Val Acc 71%, Loss 0.591098]
Iter 770. [Val Acc 72%, Loss 0.579618]
Iter 780. [Val Acc 72%, Loss 0.581651]
Iter 790. [Val Acc 72%, Loss 0.584673]
Iter 800. [Val Acc 72%, Loss 0.590660]
Iter 810. [Val Acc 71%, Loss 0.594152]
Iter 820. [Val Acc 70%, Loss 0.595766]
Iter 830. [Val Acc 71%, Loss 0.585496]
Iter 840. [Val Acc 72%, Loss 0.580777]
Iter 850. [Val Acc 70%, Loss 0.592646]
Iter 860. [Val Acc 72%, Loss 0.581417]
Iter 870. [Val Acc 72%, Loss 0.580703]
Iter 880. [Val Acc 72%, Loss 0.585556]
Iter 890. [Val Acc 72%, Loss 0.575730]
Iter 900. [Val Acc 71%, Loss 0.603648]
Iter 910. [Val Acc 72%, Loss 0.587088]
Iter 920. [Val Acc 72%, Loss 0.579448]
Iter 930. [Val Acc 71%, Loss 0.597108]
Iter 940. [Val Acc 71%, Loss 0.594602]
Iter 950. [Val Acc 70%, Loss 0.611177]
Iter 960. [Val Acc 70%, Loss 0.606051]
Iter 970. [Val Acc 72%, Loss 0.580752]
Iter 980. [Val Acc 72%, Loss 0.580853]
Iter 990. [Val Acc 72%, Loss 0.576678]
Iter 1000. [Val Acc 72%, Loss 0.585513]
-------------------------
mu= 0.5 batch_size= 64 :
Iter 10. [Val Acc 70%, Loss 0.606173]
Iter 20. [Val Acc 72%, Loss 0.577952]
Iter 30. [Val Acc 72%, Loss 0.588444]
Iter 40. [Val Acc 72%, Loss 0.594655]
Iter 50. [Val Acc 72%, Loss 0.588764]
Iter 60. [Val Acc 72%, Loss 0.584706]
Iter 70. [Val Acc 73%, Loss 0.575156]
Iter 80. [Val Acc 72%, Loss 0.583978]
Iter 90. [Val Acc 72%, Loss 0.584858]
Iter 100. [Val Acc 72%, Loss 0.576421]
Iter 110. [Val Acc 72%, Loss 0.585372]
Iter 120. [Val Acc 72%, Loss 0.584066]
Iter 130. [Val Acc 72%, Loss 0.587507]
Iter 140. [Val Acc 71%, Loss 0.595414]
Iter 150. [Val Acc 72%, Loss 0.579779]
Iter 160. [Val Acc 73%, Loss 0.575165]
Iter 170. [Val Acc 73%, Loss 0.575980]
Iter 180. [Val Acc 72%, Loss 0.573750]
Iter 190. [Val Acc 73%, Loss 0.576006]
Iter 200. [Val Acc 71%, Loss 0.588216]
Iter 210. [Val Acc 72%, Loss 0.572224]
Iter 220. [Val Acc 72%, Loss 0.591985]
Iter 230. [Val Acc 72%, Loss 0.583819]
Iter 240. [Val Acc 70%, Loss 0.606819]
Iter 250. [Val Acc 72%, Loss 0.577677]
Iter 260. [Val Acc 73%, Loss 0.573635]
Iter 270. [Val Acc 72%, Loss 0.588160]
Iter 280. [Val Acc 72%, Loss 0.581591]
Iter 290. [Val Acc 73%, Loss 0.577572]
Iter 300. [Val Acc 73%, Loss 0.571340]
Iter 310. [Val Acc 72%, Loss 0.583058]
Iter 320. [Val Acc 71%, Loss 0.598982]
Iter 330. [Val Acc 72%, Loss 0.582725]
Iter 340. [Val Acc 72%, Loss 0.586177]
Iter 350. [Val Acc 71%, Loss 0.595499]
Iter 360. [Val Acc 71%, Loss 0.599800]
Iter 370. [Val Acc 70%, Loss 0.636335]
Iter 380. [Val Acc 71%, Loss 0.590918]
Iter 390. [Val Acc 72%, Loss 0.575994]
Iter 400. [Val Acc 72%, Loss 0.578398]
Iter 410. [Val Acc 71%, Loss 0.589907]
Iter 420. [Val Acc 72%, Loss 0.591513]
Iter 430. [Val Acc 72%, Loss 0.581741]
Iter 440. [Val Acc 69%, Loss 0.629751]
Iter 450. [Val Acc 73%, Loss 0.572014]
Iter 460. [Val Acc 71%, Loss 0.582875]
Iter 470. [Val Acc 71%, Loss 0.586935]
Iter 480. [Val Acc 72%, Loss 0.575920]
Iter 490. [Val Acc 72%, Loss 0.586978]
Iter 500. [Val Acc 73%, Loss 0.573966]
Iter 510. [Val Acc 73%, Loss 0.576928]
Iter 520. [Val Acc 72%, Loss 0.579794]
Iter 530. [Val Acc 70%, Loss 0.594891]
Iter 540. [Val Acc 73%, Loss 0.567873]
Iter 550. [Val Acc 71%, Loss 0.597763]
Iter 560. [Val Acc 72%, Loss 0.586892]
Iter 570. [Val Acc 69%, Loss 0.660220]
Iter 580. [Val Acc 72%, Loss 0.591923]
Iter 590. [Val Acc 72%, Loss 0.584776]
Iter 600. [Val Acc 72%, Loss 0.583607]
Iter 610. [Val Acc 72%, Loss 0.586966]
Iter 620. [Val Acc 72%, Loss 0.581606]
Iter 630. [Val Acc 72%, Loss 0.589886]
Iter 640. [Val Acc 73%, Loss 0.572823]
Iter 650. [Val Acc 71%, Loss 0.597025]
Iter 660. [Val Acc 72%, Loss 0.584365]
Iter 670. [Val Acc 72%, Loss 0.591095]
Iter 680. [Val Acc 72%, Loss 0.601383]
Iter 690. [Val Acc 72%, Loss 0.581686]
Iter 700. [Val Acc 73%, Loss 0.579637]
Iter 710. [Val Acc 72%, Loss 0.585211]
Iter 720. [Val Acc 73%, Loss 0.578331]
Iter 730. [Val Acc 73%, Loss 0.574596]
Iter 740. [Val Acc 72%, Loss 0.582471]
Iter 750. [Val Acc 71%, Loss 0.600619]
Iter 760. [Val Acc 72%, Loss 0.575043]
Iter 770. [Val Acc 73%, Loss 0.582886]
Iter 780. [Val Acc 73%, Loss 0.589236]
Iter 790. [Val Acc 73%, Loss 0.583253]
Iter 800. [Val Acc 71%, Loss 0.603794]
Iter 810. [Val Acc 73%, Loss 0.591337]
Iter 820. [Val Acc 71%, Loss 0.588915]
Iter 830. [Val Acc 71%, Loss 0.607162]
Iter 840. [Val Acc 72%, Loss 0.579243]
Iter 850. [Val Acc 72%, Loss 0.579465]
Iter 860. [Val Acc 71%, Loss 0.593722]
Iter 870. [Val Acc 72%, Loss 0.590360]
Iter 880. [Val Acc 70%, Loss 0.619208]
Iter 890. [Val Acc 72%, Loss 0.579833]
Iter 900. [Val Acc 72%, Loss 0.581384]
Iter 910. [Val Acc 72%, Loss 0.581583]
Iter 920. [Val Acc 72%, Loss 0.586999]
Iter 930. [Val Acc 72%, Loss 0.587639]
Iter 940. [Val Acc 71%, Loss 0.590603]
Iter 950. [Val Acc 72%, Loss 0.578699]
Iter 960. [Val Acc 72%, Loss 0.599932]
Iter 970. [Val Acc 73%, Loss 0.573691]
Iter 980. [Val Acc 72%, Loss 0.588930]
Iter 990. [Val Acc 72%, Loss 0.589555]
Iter 1000. [Val Acc 71%, Loss 0.591204]
-------------------------
mu= 0.5 batch_size= 1024 :
Iter 10. [Val Acc 71%, Loss 0.589210]
Iter 20. [Val Acc 72%, Loss 0.603029]
Iter 30. [Val Acc 65%, Loss 0.744261]
Iter 40. [Val Acc 71%, Loss 0.602223]
Iter 50. [Val Acc 72%, Loss 0.590086]
Iter 60. [Val Acc 73%, Loss 0.583704]
Iter 70. [Val Acc 72%, Loss 0.587972]
Iter 80. [Val Acc 73%, Loss 0.577430]
Iter 90. [Val Acc 73%, Loss 0.569298]
Iter 100. [Val Acc 71%, Loss 0.598593]
Iter 110. [Val Acc 73%, Loss 0.581486]
Iter 120. [Val Acc 72%, Loss 0.599007]
Iter 130. [Val Acc 72%, Loss 0.575619]
Iter 140. [Val Acc 71%, Loss 0.591104]
Iter 150. [Val Acc 72%, Loss 0.580432]
Iter 160. [Val Acc 72%, Loss 0.580477]
Iter 170. [Val Acc 72%, Loss 0.581285]
Iter 180. [Val Acc 72%, Loss 0.591494]
Iter 190. [Val Acc 71%, Loss 0.582207]
Iter 200. [Val Acc 72%, Loss 0.586299]
Iter 210. [Val Acc 72%, Loss 0.599889]
Iter 220. [Val Acc 72%, Loss 0.580448]
Iter 230. [Val Acc 72%, Loss 0.580767]
Iter 240. [Val Acc 73%, Loss 0.577602]
Iter 250. [Val Acc 71%, Loss 0.600360]
Iter 260. [Val Acc 72%, Loss 0.590913]
Iter 270. [Val Acc 72%, Loss 0.589140]
Iter 280. [Val Acc 71%, Loss 0.593471]
Iter 290. [Val Acc 72%, Loss 0.578113]
Iter 300. [Val Acc 72%, Loss 0.572821]
Iter 310. [Val Acc 72%, Loss 0.591579]
Iter 320. [Val Acc 73%, Loss 0.582045]
Iter 330. [Val Acc 73%, Loss 0.576280]
Iter 340. [Val Acc 72%, Loss 0.579601]
Iter 350. [Val Acc 71%, Loss 0.605440]
Iter 360. [Val Acc 72%, Loss 0.584608]
Iter 370. [Val Acc 72%, Loss 0.582736]
Iter 380. [Val Acc 72%, Loss 0.575237]
Iter 390. [Val Acc 72%, Loss 0.575676]
Iter 400. [Val Acc 71%, Loss 0.608766]
Iter 410. [Val Acc 73%, Loss 0.576581]
Iter 420. [Val Acc 72%, Loss 0.581078]
Iter 430. [Val Acc 71%, Loss 0.594714]
Iter 440. [Val Acc 72%, Loss 0.583900]
Iter 450. [Val Acc 69%, Loss 0.656565]
Iter 460. [Val Acc 71%, Loss 0.589886]
Iter 470. [Val Acc 71%, Loss 0.600099]
Iter 480. [Val Acc 71%, Loss 0.606505]
Iter 490. [Val Acc 71%, Loss 0.593084]
Iter 500. [Val Acc 72%, Loss 0.580861]
Iter 510. [Val Acc 72%, Loss 0.577896]
Iter 520. [Val Acc 72%, Loss 0.586358]
Iter 530. [Val Acc 73%, Loss 0.575788]
Iter 540. [Val Acc 72%, Loss 0.587966]
Iter 550. [Val Acc 72%, Loss 0.589470]
Iter 560. [Val Acc 72%, Loss 0.587236]
Iter 570. [Val Acc 72%, Loss 0.592914]
Iter 580. [Val Acc 72%, Loss 0.586646]
Iter 590. [Val Acc 72%, Loss 0.583243]
Iter 600. [Val Acc 72%, Loss 0.583611]
Iter 610. [Val Acc 72%, Loss 0.587218]
Iter 620. [Val Acc 70%, Loss 0.601483]
Iter 630. [Val Acc 71%, Loss 0.592276]
Iter 640. [Val Acc 72%, Loss 0.586130]
Iter 650. [Val Acc 71%, Loss 0.585050]
Iter 660. [Val Acc 71%, Loss 0.600727]
Iter 670. [Val Acc 72%, Loss 0.592090]
Iter 680. [Val Acc 71%, Loss 0.590011]
Iter 690. [Val Acc 73%, Loss 0.575244]
Iter 700. [Val Acc 72%, Loss 0.585503]
Iter 710. [Val Acc 72%, Loss 0.593004]
Iter 720. [Val Acc 71%, Loss 0.586811]
Iter 730. [Val Acc 70%, Loss 0.604519]
Iter 740. [Val Acc 72%, Loss 0.579787]
Iter 750. [Val Acc 72%, Loss 0.578156]
Iter 760. [Val Acc 71%, Loss 0.584004]
Iter 770. [Val Acc 72%, Loss 0.575973]
Iter 780. [Val Acc 72%, Loss 0.580794]
Iter 790. [Val Acc 71%, Loss 0.594087]
Iter 800. [Val Acc 72%, Loss 0.585583]
Iter 810. [Val Acc 72%, Loss 0.591155]
Iter 820. [Val Acc 72%, Loss 0.582567]
Iter 830. [Val Acc 71%, Loss 0.603483]
Iter 840. [Val Acc 72%, Loss 0.575608]
Iter 850. [Val Acc 72%, Loss 0.581330]
Iter 860. [Val Acc 73%, Loss 0.575886]
Iter 870. [Val Acc 72%, Loss 0.584772]
Iter 880. [Val Acc 72%, Loss 0.583165]
Iter 890. [Val Acc 72%, Loss 0.582396]
Iter 900. [Val Acc 72%, Loss 0.582213]
Iter 910. [Val Acc 72%, Loss 0.586524]
Iter 920. [Val Acc 72%, Loss 0.584792]
Iter 930. [Val Acc 71%, Loss 0.593124]
Iter 940. [Val Acc 71%, Loss 0.587475]
Iter 950. [Val Acc 72%, Loss 0.577246]
Iter 960. [Val Acc 72%, Loss 0.584198]
Iter 970. [Val Acc 72%, Loss 0.591193]
Iter 980. [Val Acc 73%, Loss 0.588141]
Iter 990. [Val Acc 72%, Loss 0.578465]
Iter 1000. [Val Acc 71%, Loss 0.600692]
-------------------------
mu= 0.5 batch_size= 50000 :
Iter 10. [Val Acc 72%, Loss 0.592683]
Iter 20. [Val Acc 72%, Loss 0.579416]
Iter 30. [Val Acc 72%, Loss 0.583694]
Iter 40. [Val Acc 73%, Loss 0.578678]
Iter 50. [Val Acc 73%, Loss 0.581302]
Iter 60. [Val Acc 72%, Loss 0.579771]
Iter 70. [Val Acc 71%, Loss 0.593267]
Iter 80. [Val Acc 72%, Loss 0.578987]
Iter 90. [Val Acc 72%, Loss 0.589618]
Iter 100. [Val Acc 71%, Loss 0.601089]
Iter 110. [Val Acc 69%, Loss 0.639295]
Iter 120. [Val Acc 72%, Loss 0.587517]
Iter 130. [Val Acc 72%, Loss 0.583127]
Iter 140. [Val Acc 72%, Loss 0.580135]
Iter 150. [Val Acc 73%, Loss 0.573530]
Iter 160. [Val Acc 71%, Loss 0.588180]
Iter 170. [Val Acc 72%, Loss 0.585047]
Iter 180. [Val Acc 72%, Loss 0.581876]
Iter 190. [Val Acc 72%, Loss 0.583647]
Iter 200. [Val Acc 72%, Loss 0.583407]
Iter 210. [Val Acc 72%, Loss 0.586119]
Iter 220. [Val Acc 72%, Loss 0.593284]
Iter 230. [Val Acc 73%, Loss 0.572868]
Iter 240. [Val Acc 72%, Loss 0.589600]
Iter 250. [Val Acc 72%, Loss 0.583290]
Iter 260. [Val Acc 72%, Loss 0.587017]
Iter 270. [Val Acc 73%, Loss 0.577588]
Iter 280. [Val Acc 72%, Loss 0.591181]
Iter 290. [Val Acc 73%, Loss 0.576520]
Iter 300. [Val Acc 72%, Loss 0.583725]
Iter 310. [Val Acc 73%, Loss 0.573857]
Iter 320. [Val Acc 72%, Loss 0.587531]
Iter 330. [Val Acc 73%, Loss 0.586787]
Iter 340. [Val Acc 70%, Loss 0.620427]
Iter 350. [Val Acc 70%, Loss 0.611747]
Iter 360. [Val Acc 73%, Loss 0.576199]
Iter 370. [Val Acc 73%, Loss 0.579737]
Iter 380. [Val Acc 73%, Loss 0.577388]
Iter 390. [Val Acc 73%, Loss 0.572786]
Iter 400. [Val Acc 71%, Loss 0.596543]
Iter 410. [Val Acc 72%, Loss 0.577185]
Iter 420. [Val Acc 71%, Loss 0.598082]
Iter 430. [Val Acc 73%, Loss 0.575929]
Iter 440. [Val Acc 72%, Loss 0.578314]
Iter 450. [Val Acc 72%, Loss 0.589949]
Iter 460. [Val Acc 72%, Loss 0.596015]
Iter 470. [Val Acc 72%, Loss 0.589569]
Iter 480. [Val Acc 72%, Loss 0.587915]
Iter 490. [Val Acc 71%, Loss 0.597030]
Iter 500. [Val Acc 72%, Loss 0.579541]
Iter 510. [Val Acc 72%, Loss 0.584864]
Iter 520. [Val Acc 72%, Loss 0.582931]
Iter 530. [Val Acc 72%, Loss 0.590454]
Iter 540. [Val Acc 71%, Loss 0.585117]
Iter 550. [Val Acc 72%, Loss 0.592542]
Iter 560. [Val Acc 72%, Loss 0.587635]
Iter 570. [Val Acc 71%, Loss 0.603857]
Iter 580. [Val Acc 72%, Loss 0.584066]
Iter 590. [Val Acc 73%, Loss 0.578845]
Iter 600. [Val Acc 72%, Loss 0.594258]
Iter 610. [Val Acc 72%, Loss 0.590250]
Iter 620. [Val Acc 72%, Loss 0.592393]
Iter 630. [Val Acc 72%, Loss 0.582076]
Iter 640. [Val Acc 71%, Loss 0.588331]
Iter 650. [Val Acc 71%, Loss 0.601237]
Iter 660. [Val Acc 71%, Loss 0.601165]
Iter 670. [Val Acc 71%, Loss 0.593265]
Iter 680. [Val Acc 71%, Loss 0.610716]
Iter 690. [Val Acc 72%, Loss 0.588713]
Iter 700. [Val Acc 73%, Loss 0.575681]
Iter 710. [Val Acc 72%, Loss 0.579087]
Iter 720. [Val Acc 73%, Loss 0.575245]
Iter 730. [Val Acc 72%, Loss 0.584133]
Iter 740. [Val Acc 72%, Loss 0.580486]
Iter 750. [Val Acc 73%, Loss 0.576796]
Iter 760. [Val Acc 71%, Loss 0.589501]
Iter 770. [Val Acc 72%, Loss 0.579829]
Iter 780. [Val Acc 72%, Loss 0.585343]
Iter 790. [Val Acc 72%, Loss 0.580408]
Iter 800. [Val Acc 73%, Loss 0.571175]
Iter 810. [Val Acc 72%, Loss 0.580470]
Iter 820. [Val Acc 72%, Loss 0.583257]
Iter 830. [Val Acc 72%, Loss 0.578053]
Iter 840. [Val Acc 73%, Loss 0.576516]
Iter 850. [Val Acc 72%, Loss 0.572452]
Iter 860. [Val Acc 72%, Loss 0.577578]
Iter 870. [Val Acc 72%, Loss 0.582210]
Iter 880. [Val Acc 72%, Loss 0.592086]
Iter 890. [Val Acc 73%, Loss 0.576531]
Iter 900. [Val Acc 73%, Loss 0.574552]
Iter 910. [Val Acc 72%, Loss 0.585882]
Iter 920. [Val Acc 72%, Loss 0.583272]
Iter 930. [Val Acc 72%, Loss 0.575140]
Iter 940. [Val Acc 72%, Loss 0.585038]
Iter 950. [Val Acc 73%, Loss 0.580774]
Iter 960. [Val Acc 71%, Loss 0.593172]
Iter 970. [Val Acc 71%, Loss 0.590174]
Iter 980. [Val Acc 72%, Loss 0.588020]
Iter 990. [Val Acc 72%, Loss 0.573353]
Iter 1000. [Val Acc 72%, Loss 0.583535]
-------------------------
```

```
mu= 1 batch_size= 1 :
Iter 10. [Val Acc 65%, Loss 0.835307]
Iter 20. [Val Acc 70%, Loss 0.624538]
Iter 30. [Val Acc 70%, Loss 0.648395]
Iter 40. [Val Acc 71%, Loss 0.611388]
Iter 50. [Val Acc 71%, Loss 0.606219]
Iter 60. [Val Acc 71%, Loss 0.622821]
Iter 70. [Val Acc 71%, Loss 0.606585]
Iter 80. [Val Acc 70%, Loss 0.636981]
Iter 90. [Val Acc 71%, Loss 0.607346]
Iter 100. [Val Acc 68%, Loss 0.674119]
Iter 110. [Val Acc 71%, Loss 0.631252]
Iter 120. [Val Acc 71%, Loss 0.606393]
Iter 130. [Val Acc 69%, Loss 0.632669]
Iter 140. [Val Acc 70%, Loss 0.626606]
Iter 150. [Val Acc 70%, Loss 0.633659]
Iter 160. [Val Acc 66%, Loss 0.704981]
Iter 170. [Val Acc 69%, Loss 0.640457]
Iter 180. [Val Acc 70%, Loss 0.638301]
Iter 190. [Val Acc 68%, Loss 0.678143]
Iter 200. [Val Acc 70%, Loss 0.635142]
Iter 210. [Val Acc 68%, Loss 0.664631]
Iter 220. [Val Acc 70%, Loss 0.632760]
Iter 230. [Val Acc 71%, Loss 0.621955]
Iter 240. [Val Acc 71%, Loss 0.628745]
Iter 250. [Val Acc 71%, Loss 0.618109]
Iter 260. [Val Acc 70%, Loss 0.646655]
Iter 270. [Val Acc 72%, Loss 0.611295]
Iter 280. [Val Acc 70%, Loss 0.642134]
Iter 290. [Val Acc 72%, Loss 0.603068]
Iter 300. [Val Acc 71%, Loss 0.612133]
Iter 310. [Val Acc 71%, Loss 0.634346]
Iter 320. [Val Acc 71%, Loss 0.627186]
Iter 330. [Val Acc 72%, Loss 0.599014]
Iter 340. [Val Acc 72%, Loss 0.599767]
Iter 350. [Val Acc 71%, Loss 0.620333]
Iter 360. [Val Acc 70%, Loss 0.640612]
Iter 370. [Val Acc 70%, Loss 0.632140]
Iter 380. [Val Acc 71%, Loss 0.605680]
Iter 390. [Val Acc 66%, Loss 0.692406]
Iter 400. [Val Acc 71%, Loss 0.618953]
Iter 410. [Val Acc 70%, Loss 0.631884]
Iter 420. [Val Acc 71%, Loss 0.610257]
Iter 430. [Val Acc 71%, Loss 0.615260]
Iter 440. [Val Acc 68%, Loss 0.674961]
Iter 450. [Val Acc 70%, Loss 0.636760]
Iter 460. [Val Acc 71%, Loss 0.622234]
Iter 470. [Val Acc 72%, Loss 0.614751]
Iter 480. [Val Acc 68%, Loss 0.679807]
Iter 490. [Val Acc 67%, Loss 0.676813]
Iter 500. [Val Acc 70%, Loss 0.633809]
Iter 510. [Val Acc 70%, Loss 0.634957]
Iter 520. [Val Acc 70%, Loss 0.677149]
Iter 530. [Val Acc 70%, Loss 0.632326]
Iter 540. [Val Acc 72%, Loss 0.598494]
Iter 550. [Val Acc 68%, Loss 0.649885]
Iter 560. [Val Acc 65%, Loss 0.747819]
Iter 570. [Val Acc 71%, Loss 0.632694]
Iter 580. [Val Acc 70%, Loss 0.623372]
Iter 590. [Val Acc 70%, Loss 0.664098]
Iter 600. [Val Acc 67%, Loss 0.690233]
Iter 610. [Val Acc 70%, Loss 0.650998]
Iter 620. [Val Acc 71%, Loss 0.624001]
Iter 630. [Val Acc 71%, Loss 0.615079]
Iter 640. [Val Acc 70%, Loss 0.623445]
Iter 650. [Val Acc 68%, Loss 0.666189]
Iter 660. [Val Acc 69%, Loss 0.665159]
Iter 670. [Val Acc 71%, Loss 0.609871]
Iter 680. [Val Acc 71%, Loss 0.627761]
Iter 690. [Val Acc 68%, Loss 0.768204]
Iter 700. [Val Acc 67%, Loss 0.682128]
Iter 710. [Val Acc 67%, Loss 0.711350]
Iter 720. [Val Acc 71%, Loss 0.615439]
Iter 730. [Val Acc 70%, Loss 0.642120]
Iter 740. [Val Acc 69%, Loss 0.666598]
Iter 750. [Val Acc 71%, Loss 0.626535]
Iter 760. [Val Acc 71%, Loss 0.621660]
Iter 770. [Val Acc 70%, Loss 0.640999]
Iter 780. [Val Acc 72%, Loss 0.608338]
Iter 790. [Val Acc 71%, Loss 0.633275]
Iter 800. [Val Acc 70%, Loss 0.626994]
Iter 810. [Val Acc 70%, Loss 0.619370]
Iter 820. [Val Acc 70%, Loss 0.632602]
Iter 830. [Val Acc 70%, Loss 0.632094]
Iter 840. [Val Acc 69%, Loss 0.631691]
Iter 850. [Val Acc 68%, Loss 0.653616]
Iter 860. [Val Acc 71%, Loss 0.644223]
Iter 870. [Val Acc 69%, Loss 0.627117]
Iter 880. [Val Acc 68%, Loss 0.663094]
Iter 890. [Val Acc 70%, Loss 0.634297]
Iter 900. [Val Acc 71%, Loss 0.625664]
Iter 910. [Val Acc 71%, Loss 0.637111]
Iter 920. [Val Acc 71%, Loss 0.627618]
Iter 930. [Val Acc 72%, Loss 0.614587]
Iter 940. [Val Acc 69%, Loss 0.648853]
Iter 950. [Val Acc 71%, Loss 0.610537]
Iter 960. [Val Acc 70%, Loss 0.645824]
Iter 970. [Val Acc 69%, Loss 0.663227]
Iter 980. [Val Acc 71%, Loss 0.609769]
Iter 990. [Val Acc 71%, Loss 0.609153]
Iter 1000. [Val Acc 71%, Loss 0.616064]
-------------------------
mu= 1 batch_size= 64 :
Iter 10. [Val Acc 70%, Loss 0.629198]
Iter 20. [Val Acc 66%, Loss 0.695369]
Iter 30. [Val Acc 71%, Loss 0.607919]
Iter 40. [Val Acc 69%, Loss 0.638659]
Iter 50. [Val Acc 67%, Loss 0.688053]
Iter 60. [Val Acc 68%, Loss 0.693892]
Iter 70. [Val Acc 71%, Loss 0.615990]
Iter 80. [Val Acc 72%, Loss 0.599219]
Iter 90. [Val Acc 71%, Loss 0.629675]
Iter 100. [Val Acc 71%, Loss 0.622717]
Iter 110. [Val Acc 69%, Loss 0.641184]
Iter 120. [Val Acc 71%, Loss 0.611580]
Iter 130. [Val Acc 69%, Loss 0.649739]
Iter 140. [Val Acc 70%, Loss 0.634307]
Iter 150. [Val Acc 69%, Loss 0.632484]
Iter 160. [Val Acc 71%, Loss 0.613134]
Iter 170. [Val Acc 69%, Loss 0.652090]
Iter 180. [Val Acc 67%, Loss 0.734856]
Iter 190. [Val Acc 71%, Loss 0.612898]
Iter 200. [Val Acc 70%, Loss 0.626283]
Iter 210. [Val Acc 70%, Loss 0.632082]
Iter 220. [Val Acc 72%, Loss 0.616310]
Iter 230. [Val Acc 71%, Loss 0.612020]
Iter 240. [Val Acc 68%, Loss 0.669219]
Iter 250. [Val Acc 71%, Loss 0.617777]
Iter 260. [Val Acc 64%, Loss 0.754732]
Iter 270. [Val Acc 70%, Loss 0.636857]
Iter 280. [Val Acc 71%, Loss 0.623842]
Iter 290. [Val Acc 72%, Loss 0.612712]
Iter 300. [Val Acc 67%, Loss 0.708748]
Iter 310. [Val Acc 71%, Loss 0.608174]
Iter 320. [Val Acc 65%, Loss 0.778001]
Iter 330. [Val Acc 71%, Loss 0.620782]
Iter 340. [Val Acc 71%, Loss 0.632840]
Iter 350. [Val Acc 71%, Loss 0.619756]
Iter 360. [Val Acc 71%, Loss 0.621645]
Iter 370. [Val Acc 72%, Loss 0.609539]
Iter 380. [Val Acc 70%, Loss 0.625794]
Iter 390. [Val Acc 71%, Loss 0.624004]
Iter 400. [Val Acc 71%, Loss 0.606671]
Iter 410. [Val Acc 70%, Loss 0.625599]
Iter 420. [Val Acc 70%, Loss 0.618382]
Iter 430. [Val Acc 71%, Loss 0.609106]
Iter 440. [Val Acc 70%, Loss 0.623340]
Iter 450. [Val Acc 71%, Loss 0.597850]
Iter 460. [Val Acc 70%, Loss 0.640124]
Iter 470. [Val Acc 70%, Loss 0.632079]
Iter 480. [Val Acc 70%, Loss 0.630415]
Iter 490. [Val Acc 70%, Loss 0.635406]
Iter 500. [Val Acc 69%, Loss 0.649818]
Iter 510. [Val Acc 71%, Loss 0.609786]
Iter 520. [Val Acc 71%, Loss 0.612525]
Iter 530. [Val Acc 70%, Loss 0.637019]
Iter 540. [Val Acc 71%, Loss 0.611348]
Iter 550. [Val Acc 70%, Loss 0.609801]
Iter 560. [Val Acc 66%, Loss 0.740150]
Iter 570. [Val Acc 71%, Loss 0.618078]
Iter 580. [Val Acc 71%, Loss 0.629268]
Iter 590. [Val Acc 71%, Loss 0.600015]
Iter 600. [Val Acc 69%, Loss 0.683220]
Iter 610. [Val Acc 68%, Loss 0.681337]
Iter 620. [Val Acc 70%, Loss 0.619363]
Iter 630. [Val Acc 70%, Loss 0.626569]
Iter 640. [Val Acc 70%, Loss 0.627416]
Iter 650. [Val Acc 70%, Loss 0.630546]
Iter 660. [Val Acc 66%, Loss 0.734672]
Iter 670. [Val Acc 66%, Loss 0.792481]
Iter 680. [Val Acc 71%, Loss 0.614539]
Iter 690. [Val Acc 68%, Loss 0.660943]
Iter 700. [Val Acc 71%, Loss 0.625004]
Iter 710. [Val Acc 71%, Loss 0.633742]
Iter 720. [Val Acc 71%, Loss 0.611698]
Iter 730. [Val Acc 70%, Loss 0.623131]
Iter 740. [Val Acc 71%, Loss 0.616610]
Iter 750. [Val Acc 70%, Loss 0.624855]
Iter 760. [Val Acc 66%, Loss 0.702875]
Iter 770. [Val Acc 68%, Loss 0.676050]
Iter 780. [Val Acc 71%, Loss 0.624335]
Iter 790. [Val Acc 70%, Loss 0.641198]
Iter 800. [Val Acc 71%, Loss 0.619213]
Iter 810. [Val Acc 67%, Loss 0.690588]
Iter 820. [Val Acc 72%, Loss 0.600636]
Iter 830. [Val Acc 72%, Loss 0.604822]
Iter 840. [Val Acc 70%, Loss 0.630850]
Iter 850. [Val Acc 70%, Loss 0.642288]
Iter 860. [Val Acc 68%, Loss 0.695363]
Iter 870. [Val Acc 69%, Loss 0.651445]
Iter 880. [Val Acc 71%, Loss 0.618260]
Iter 890. [Val Acc 67%, Loss 0.701989]
Iter 900. [Val Acc 72%, Loss 0.610612]
Iter 910. [Val Acc 72%, Loss 0.616255]
Iter 920. [Val Acc 70%, Loss 0.635534]
Iter 930. [Val Acc 70%, Loss 0.633450]
Iter 940. [Val Acc 68%, Loss 0.706726]
Iter 950. [Val Acc 70%, Loss 0.636933]
Iter 960. [Val Acc 62%, Loss 0.797707]
Iter 970. [Val Acc 72%, Loss 0.625017]
Iter 980. [Val Acc 70%, Loss 0.638366]
Iter 990. [Val Acc 70%, Loss 0.629172]
Iter 1000. [Val Acc 71%, Loss 0.604121]
-------------------------
mu= 1 batch_size= 1024 :
Iter 10. [Val Acc 68%, Loss 0.649252]
Iter 20. [Val Acc 71%, Loss 0.611786]
Iter 30. [Val Acc 71%, Loss 0.610930]
Iter 40. [Val Acc 70%, Loss 0.627276]
Iter 50. [Val Acc 65%, Loss 0.758272]
Iter 60. [Val Acc 71%, Loss 0.612275]
Iter 70. [Val Acc 70%, Loss 0.626646]
Iter 80. [Val Acc 70%, Loss 0.612988]
Iter 90. [Val Acc 71%, Loss 0.604877]
Iter 100. [Val Acc 69%, Loss 0.645123]
Iter 110. [Val Acc 70%, Loss 0.613425]
Iter 120. [Val Acc 71%, Loss 0.613258]
Iter 130. [Val Acc 70%, Loss 0.612942]
Iter 140. [Val Acc 67%, Loss 0.662790]
Iter 150. [Val Acc 68%, Loss 0.672127]
Iter 160. [Val Acc 70%, Loss 0.629385]
Iter 170. [Val Acc 71%, Loss 0.606426]
Iter 180. [Val Acc 70%, Loss 0.630142]
Iter 190. [Val Acc 71%, Loss 0.621830]
Iter 200. [Val Acc 71%, Loss 0.617339]
Iter 210. [Val Acc 69%, Loss 0.643586]
Iter 220. [Val Acc 72%, Loss 0.598228]
Iter 230. [Val Acc 69%, Loss 0.648760]
Iter 240. [Val Acc 72%, Loss 0.609973]
Iter 250. [Val Acc 68%, Loss 0.663813]
Iter 260. [Val Acc 71%, Loss 0.627936]
Iter 270. [Val Acc 71%, Loss 0.603384]
Iter 280. [Val Acc 70%, Loss 0.624479]
Iter 290. [Val Acc 69%, Loss 0.659565]
Iter 300. [Val Acc 70%, Loss 0.623267]
Iter 310. [Val Acc 69%, Loss 0.682700]
Iter 320. [Val Acc 71%, Loss 0.612237]
Iter 330. [Val Acc 63%, Loss 0.782602]
Iter 340. [Val Acc 70%, Loss 0.622531]
Iter 350. [Val Acc 66%, Loss 0.769405]
Iter 360. [Val Acc 70%, Loss 0.641148]
Iter 370. [Val Acc 71%, Loss 0.626583]
Iter 380. [Val Acc 71%, Loss 0.612345]
Iter 390. [Val Acc 70%, Loss 0.630485]
Iter 400. [Val Acc 72%, Loss 0.603227]
Iter 410. [Val Acc 70%, Loss 0.643332]
Iter 420. [Val Acc 69%, Loss 0.701056]
Iter 430. [Val Acc 70%, Loss 0.639313]
Iter 440. [Val Acc 70%, Loss 0.675662]
Iter 450. [Val Acc 71%, Loss 0.611969]
Iter 460. [Val Acc 71%, Loss 0.630756]
Iter 470. [Val Acc 68%, Loss 0.702735]
Iter 480. [Val Acc 71%, Loss 0.608782]
Iter 490. [Val Acc 72%, Loss 0.613070]
Iter 500. [Val Acc 69%, Loss 0.652675]
Iter 510. [Val Acc 70%, Loss 0.621619]
Iter 520. [Val Acc 70%, Loss 0.645772]
Iter 530. [Val Acc 70%, Loss 0.617525]
Iter 540. [Val Acc 70%, Loss 0.636999]
Iter 550. [Val Acc 67%, Loss 0.705442]
Iter 560. [Val Acc 71%, Loss 0.609372]
Iter 570. [Val Acc 71%, Loss 0.624787]
Iter 580. [Val Acc 63%, Loss 0.909281]
Iter 590. [Val Acc 69%, Loss 0.660922]
Iter 600. [Val Acc 70%, Loss 0.629545]
Iter 610. [Val Acc 70%, Loss 0.633079]
Iter 620. [Val Acc 71%, Loss 0.622469]
Iter 630. [Val Acc 70%, Loss 0.621392]
Iter 640. [Val Acc 70%, Loss 0.630855]
Iter 650. [Val Acc 71%, Loss 0.605529]
Iter 660. [Val Acc 71%, Loss 0.611042]
Iter 670. [Val Acc 71%, Loss 0.610900]
Iter 680. [Val Acc 69%, Loss 0.674227]
Iter 690. [Val Acc 72%, Loss 0.603755]
Iter 700. [Val Acc 70%, Loss 0.643773]
Iter 710. [Val Acc 70%, Loss 0.640813]
Iter 720. [Val Acc 71%, Loss 0.632316]
Iter 730. [Val Acc 70%, Loss 0.630254]
Iter 740. [Val Acc 70%, Loss 0.649163]
Iter 750. [Val Acc 71%, Loss 0.649936]
Iter 760. [Val Acc 70%, Loss 0.640333]
Iter 770. [Val Acc 70%, Loss 0.636234]
Iter 780. [Val Acc 69%, Loss 0.672229]
Iter 790. [Val Acc 70%, Loss 0.638844]
Iter 800. [Val Acc 71%, Loss 0.634575]
Iter 810. [Val Acc 71%, Loss 0.601655]
Iter 820. [Val Acc 70%, Loss 0.632649]
Iter 830. [Val Acc 72%, Loss 0.606974]
Iter 840. [Val Acc 71%, Loss 0.628278]
Iter 850. [Val Acc 67%, Loss 0.736416]
Iter 860. [Val Acc 71%, Loss 0.610310]
Iter 870. [Val Acc 70%, Loss 0.631057]
Iter 880. [Val Acc 66%, Loss 0.721716]
Iter 890. [Val Acc 70%, Loss 0.629276]
Iter 900. [Val Acc 71%, Loss 0.611361]
Iter 910. [Val Acc 71%, Loss 0.623801]
Iter 920. [Val Acc 70%, Loss 0.612657]
Iter 930. [Val Acc 71%, Loss 0.617298]
Iter 940. [Val Acc 70%, Loss 0.633442]
Iter 950. [Val Acc 68%, Loss 0.693318]
Iter 960. [Val Acc 67%, Loss 0.728475]
Iter 970. [Val Acc 71%, Loss 0.628805]
Iter 980. [Val Acc 72%, Loss 0.613488]
Iter 990. [Val Acc 70%, Loss 0.662641]
Iter 1000. [Val Acc 70%, Loss 0.650241]
-------------------------
mu= 1 batch_size= 50000 :
Iter 10. [Val Acc 71%, Loss 0.626459]
Iter 20. [Val Acc 70%, Loss 0.663622]
Iter 30. [Val Acc 70%, Loss 0.630805]
Iter 40. [Val Acc 70%, Loss 0.624077]
Iter 50. [Val Acc 70%, Loss 0.627647]
Iter 60. [Val Acc 71%, Loss 0.636453]
Iter 70. [Val Acc 71%, Loss 0.615262]
Iter 80. [Val Acc 70%, Loss 0.622871]
Iter 90. [Val Acc 69%, Loss 0.647021]
Iter 100. [Val Acc 63%, Loss 0.914931]
Iter 110. [Val Acc 67%, Loss 0.679086]
Iter 120. [Val Acc 69%, Loss 0.648313]
Iter 130. [Val Acc 69%, Loss 0.663981]
Iter 140. [Val Acc 69%, Loss 0.696844]
Iter 150. [Val Acc 71%, Loss 0.624825]
Iter 160. [Val Acc 71%, Loss 0.617030]
Iter 170. [Val Acc 70%, Loss 0.633178]
Iter 180. [Val Acc 69%, Loss 0.628524]
Iter 190. [Val Acc 70%, Loss 0.649084]
Iter 200. [Val Acc 70%, Loss 0.623422]
Iter 210. [Val Acc 71%, Loss 0.609200]
Iter 220. [Val Acc 70%, Loss 0.627624]
Iter 230. [Val Acc 70%, Loss 0.664023]
Iter 240. [Val Acc 71%, Loss 0.620402]
Iter 250. [Val Acc 71%, Loss 0.629401]
Iter 260. [Val Acc 70%, Loss 0.624368]
Iter 270. [Val Acc 71%, Loss 0.628890]
Iter 280. [Val Acc 71%, Loss 0.610450]
Iter 290. [Val Acc 70%, Loss 0.627236]
Iter 300. [Val Acc 69%, Loss 0.651140]
Iter 310. [Val Acc 68%, Loss 0.664734]
Iter 320. [Val Acc 70%, Loss 0.618743]
Iter 330. [Val Acc 69%, Loss 0.637422]
Iter 340. [Val Acc 66%, Loss 0.753548]
Iter 350. [Val Acc 69%, Loss 0.666854]
Iter 360. [Val Acc 71%, Loss 0.634307]
Iter 370. [Val Acc 71%, Loss 0.614273]
Iter 380. [Val Acc 71%, Loss 0.610944]
Iter 390. [Val Acc 71%, Loss 0.609151]
Iter 400. [Val Acc 69%, Loss 0.647926]
Iter 410. [Val Acc 70%, Loss 0.637192]
Iter 420. [Val Acc 70%, Loss 0.625743]
Iter 430. [Val Acc 69%, Loss 0.642357]
Iter 440. [Val Acc 71%, Loss 0.619092]
Iter 450. [Val Acc 71%, Loss 0.624259]
Iter 460. [Val Acc 72%, Loss 0.599912]
Iter 470. [Val Acc 69%, Loss 0.647424]
Iter 480. [Val Acc 70%, Loss 0.638805]
Iter 490. [Val Acc 72%, Loss 0.597873]
Iter 500. [Val Acc 71%, Loss 0.633345]
Iter 510. [Val Acc 72%, Loss 0.613755]
Iter 520. [Val Acc 69%, Loss 0.635977]
Iter 530. [Val Acc 69%, Loss 0.668317]
Iter 540. [Val Acc 71%, Loss 0.619044]
Iter 550. [Val Acc 71%, Loss 0.609404]
Iter 560. [Val Acc 70%, Loss 0.631507]
Iter 570. [Val Acc 71%, Loss 0.613583]
Iter 580. [Val Acc 71%, Loss 0.611281]
Iter 590. [Val Acc 70%, Loss 0.636085]
Iter 600. [Val Acc 69%, Loss 0.644006]
Iter 610. [Val Acc 70%, Loss 0.650189]
Iter 620. [Val Acc 71%, Loss 0.647499]
Iter 630. [Val Acc 70%, Loss 0.660596]
Iter 640. [Val Acc 72%, Loss 0.622102]
Iter 650. [Val Acc 71%, Loss 0.607361]
Iter 660. [Val Acc 69%, Loss 0.653456]
Iter 670. [Val Acc 71%, Loss 0.626598]
Iter 680. [Val Acc 72%, Loss 0.608227]
Iter 690. [Val Acc 71%, Loss 0.612732]
Iter 700. [Val Acc 71%, Loss 0.621411]
Iter 710. [Val Acc 70%, Loss 0.645016]
Iter 720. [Val Acc 70%, Loss 0.636396]
Iter 730. [Val Acc 71%, Loss 0.635381]
Iter 740. [Val Acc 69%, Loss 0.656871]
Iter 750. [Val Acc 71%, Loss 0.616633]
Iter 760. [Val Acc 71%, Loss 0.627937]
Iter 770. [Val Acc 70%, Loss 0.628213]
Iter 780. [Val Acc 72%, Loss 0.610602]
Iter 790. [Val Acc 69%, Loss 0.667862]
Iter 800. [Val Acc 70%, Loss 0.633817]
Iter 810. [Val Acc 70%, Loss 0.640995]
Iter 820. [Val Acc 71%, Loss 0.607677]
Iter 830. [Val Acc 71%, Loss 0.621537]
Iter 840. [Val Acc 69%, Loss 0.637984]
Iter 850. [Val Acc 71%, Loss 0.645876]
Iter 860. [Val Acc 66%, Loss 0.776783]
Iter 870. [Val Acc 70%, Loss 0.620939]
Iter 880. [Val Acc 71%, Loss 0.621915]
Iter 890. [Val Acc 71%, Loss 0.614946]
Iter 900. [Val Acc 70%, Loss 0.652420]
Iter 910. [Val Acc 69%, Loss 0.652943]
Iter 920. [Val Acc 70%, Loss 0.617375]
Iter 930. [Val Acc 69%, Loss 0.675866]
Iter 940. [Val Acc 70%, Loss 0.624393]
Iter 950. [Val Acc 68%, Loss 0.700144]
Iter 960. [Val Acc 67%, Loss 0.682843]
Iter 970. [Val Acc 70%, Loss 0.628097]
Iter 980. [Val Acc 70%, Loss 0.639787]
Iter 990. [Val Acc 72%, Loss 0.619058]
Iter 1000. [Val Acc 71%, Loss 0.630849]
-------------------------
optimal mu= 0.1 optimal batch_size= 50000
```

We found the optimal weights and biases by scanning over a range of different learning rates and batch sizes for 1000 iterations each and then picking the weights and biases in which the loss over the validation set was minimal. The best weights and biases were observed when $\mu = 0.1$ and batch_size = 50000 (the size of the training data set). The best results are achieved when the batch size is maximal. The reason is that the number of iterations is the number of update steps. Because this number is constant (1000), we have the same number of updates independently of the batch size. Thus, when the batch size is large, as we learned in the lectures, we get a better approximation of the true gradient, and hence the results are better. It is important to mention that we should not conclude that the best choice is always picking the largest batch size. The reason is that if we measure the validation loss per epoch (for a fixed number of epochs), the number of update steps with a large batch size will be smaller than the number of updates when the batch size is smaller which may lead to worse performance. Furthermore, a smaller batch size can assist the algorithm in avoiding it from stuck in a local minimum or a saddle point due to the noisier gradient.

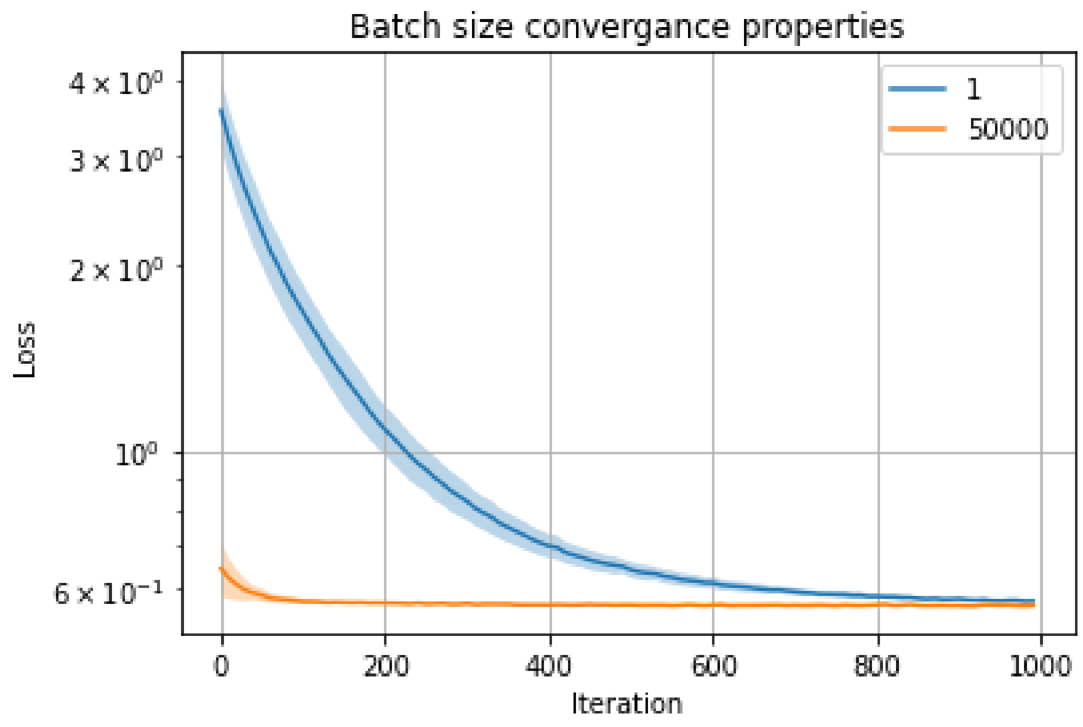
[ ]:
```
# Write your code here
mu = 0.1
batch_sizes = [1,50000]
Iterations = 1000
plt.figure()
MC_SIZE = 3
losses_vec = np.zeros((MC_SIZE, len(batch_sizes), int(Iterations/10)))
```

```python
for mc_indx in range(MC_SIZE):
    w0 = np.random.randn(90)
    b0 = np.random.randn(1)[0]
    print("monte_carlo:", mc_indx)
    for curr_idx, minibatch_size in enumerate(batch_sizes):
      print("mu=",mu, 'batch_size=', minibatch_size, ':')
      w,b,losses=run_gradient_descent(w0,b0,mu, max_iters=Iterations)
      losses_vec[mc_indx][curr_idx] = losses
      print("-------------------------")


mean = np.mean(losses_vec,axis=0)
std = np.std(losses_vec,axis=0)
for curr_idx, minibatch_size in enumerate(batch_sizes):
  plt.semilogy(range(0,Iterations,10),mean[curr_idx], label=f'{minibatch_size}')
  y_neg_error = mean[curr_idx]-std[curr_idx]
  y_pos_error = mean[curr_idx]+std[curr_idx]
  plt.fill_between(range(0,Iterations,10), y_neg_error, y_pos_error, alpha=0.3)
plt.legend(loc='best')
plt.xlabel("Iteration")
plt.ylabel("Loss")
plt.grid()
plt.title("Batch size convergance properties")
```

```
monte_carlo: 0
mu= 0.1 batch_size= 1 :
Iter 10. [Val Acc 50%, Loss 2.988975]
Iter 20. [Val Acc 50%, Loss 2.667662]
Iter 30. [Val Acc 51%, Loss 2.407711]
Iter 40. [Val Acc 52%, Loss 2.213487]
Iter 50. [Val Acc 53%, Loss 2.034886]
Iter 60. [Val Acc 54%, Loss 1.918314]
Iter 70. [Val Acc 55%, Loss 1.783832]
Iter 80. [Val Acc 55%, Loss 1.679106]
Iter 90. [Val Acc 56%, Loss 1.575458]
Iter 100. [Val Acc 57%, Loss 1.505344]
Iter 110. [Val Acc 57%, Loss 1.429909]
Iter 120. [Val Acc 58%, Loss 1.359682]
Iter 130. [Val Acc 59%, Loss 1.299835]
Iter 140. [Val Acc 59%, Loss 1.230887]
Iter 150. [Val Acc 60%, Loss 1.174670]
Iter 160. [Val Acc 61%, Loss 1.117459]
Iter 170. [Val Acc 61%, Loss 1.073838]
Iter 180. [Val Acc 62%, Loss 1.035060]
Iter 190. [Val Acc 62%, Loss 1.000929]
Iter 200. [Val Acc 63%, Loss 0.972118]
Iter 210. [Val Acc 63%, Loss 0.943865]
Iter 220. [Val Acc 63%, Loss 0.914478]
Iter 230. [Val Acc 64%, Loss 0.887211]
Iter 240. [Val Acc 64%, Loss 0.860926]
Iter 250. [Val Acc 64%, Loss 0.843443]
Iter 260. [Val Acc 65%, Loss 0.830494]
Iter 270. [Val Acc 65%, Loss 0.802618]
Iter 280. [Val Acc 66%, Loss 0.792238]
Iter 290. [Val Acc 66%, Loss 0.772187]
Iter 300. [Val Acc 66%, Loss 0.760490]
Iter 310. [Val Acc 67%, Loss 0.752141]
Iter 320. [Val Acc 67%, Loss 0.737006]
Iter 330. [Val Acc 67%, Loss 0.726057]
Iter 340. [Val Acc 68%, Loss 0.713343]
Iter 350. [Val Acc 68%, Loss 0.703581]
Iter 360. [Val Acc 68%, Loss 0.694631]
Iter 370. [Val Acc 68%, Loss 0.686823]
Iter 380. [Val Acc 69%, Loss 0.677702]
Iter 390. [Val Acc 69%, Loss 0.670663]
Iter 400. [Val Acc 69%, Loss 0.662769]
Iter 410. [Val Acc 69%, Loss 0.657274]
Iter 420. [Val Acc 69%, Loss 0.652371]
Iter 430. [Val Acc 69%, Loss 0.646517]
Iter 440. [Val Acc 69%, Loss 0.641465]
Iter 450. [Val Acc 69%, Loss 0.637293]
Iter 460. [Val Acc 70%, Loss 0.630870]
Iter 470. [Val Acc 70%, Loss 0.627512]
Iter 480. [Val Acc 70%, Loss 0.626387]
Iter 490. [Val Acc 70%, Loss 0.622051]
Iter 500. [Val Acc 70%, Loss 0.626211]
Iter 510. [Val Acc 70%, Loss 0.615420]
Iter 520. [Val Acc 71%, Loss 0.611164]
Iter 530. [Val Acc 71%, Loss 0.608687]
Iter 540. [Val Acc 71%, Loss 0.610778]
Iter 550. [Val Acc 71%, Loss 0.603462]
Iter 560. [Val Acc 71%, Loss 0.600741]
Iter 570. [Val Acc 71%, Loss 0.597940]
Iter 580. [Val Acc 71%, Loss 0.595799]
Iter 590. [Val Acc 71%, Loss 0.594981]
Iter 600. [Val Acc 71%, Loss 0.594292]
Iter 610. [Val Acc 71%, Loss 0.591960]
Iter 620. [Val Acc 72%, Loss 0.589173]
Iter 630. [Val Acc 72%, Loss 0.588827]
Iter 640. [Val Acc 72%, Loss 0.586945]
Iter 650. [Val Acc 72%, Loss 0.585873]
Iter 660. [Val Acc 72%, Loss 0.584911]
Iter 670. [Val Acc 72%, Loss 0.581751]
Iter 680. [Val Acc 72%, Loss 0.583727]
Iter 690. [Val Acc 71%, Loss 0.584256]
Iter 700. [Val Acc 72%, Loss 0.581169]
Iter 710. [Val Acc 72%, Loss 0.579066]
Iter 720. [Val Acc 72%, Loss 0.577616]
Iter 730. [Val Acc 72%, Loss 0.578711]
Iter 740. [Val Acc 72%, Loss 0.576714]
Iter 750. [Val Acc 72%, Loss 0.576335]
Iter 760. [Val Acc 72%, Loss 0.575805]
Iter 770. [Val Acc 73%, Loss 0.573263]
Iter 780. [Val Acc 73%, Loss 0.572973]
Iter 790. [Val Acc 72%, Loss 0.573444]
Iter 800. [Val Acc 72%, Loss 0.573600]
Iter 810. [Val Acc 72%, Loss 0.574825]
Iter 820. [Val Acc 72%, Loss 0.573456]
Iter 830. [Val Acc 73%, Loss 0.571975]
Iter 840. [Val Acc 72%, Loss 0.571336]
Iter 850. [Val Acc 72%, Loss 0.573646]
Iter 860. [Val Acc 73%, Loss 0.570579]
Iter 870. [Val Acc 73%, Loss 0.570091]
Iter 880. [Val Acc 72%, Loss 0.572335]
Iter 890. [Val Acc 73%, Loss 0.571174]
Iter 900. [Val Acc 72%, Loss 0.572811]
Iter 910. [Val Acc 73%, Loss 0.572232]
Iter 920. [Val Acc 73%, Loss 0.570569]
Iter 930. [Val Acc 72%, Loss 0.569783]
Iter 940. [Val Acc 73%, Loss 0.569087]
Iter 950. [Val Acc 73%, Loss 0.568024]
Iter 960. [Val Acc 73%, Loss 0.567501]
Iter 970. [Val Acc 73%, Loss 0.567914]
Iter 980. [Val Acc 73%, Loss 0.566466]
Iter 990. [Val Acc 73%, Loss 0.567172]
Iter 1000. [Val Acc 73%, Loss 0.566885]
-------------------------
mu= 0.1 batch_size= 50000 :
Iter 10. [Val Acc 69%, Loss 0.609053]
Iter 20. [Val Acc 71%, Loss 0.594825]
Iter 30. [Val Acc 71%, Loss 0.586232]
Iter 40. [Val Acc 72%, Loss 0.580912]
Iter 50. [Val Acc 72%, Loss 0.579176]
Iter 60. [Val Acc 72%, Loss 0.575480]
Iter 70. [Val Acc 73%, Loss 0.568668]
Iter 80. [Val Acc 73%, Loss 0.568389]
Iter 90. [Val Acc 73%, Loss 0.567299]
Iter 100. [Val Acc 73%, Loss 0.565149]
Iter 110. [Val Acc 73%, Loss 0.565355]
Iter 120. [Val Acc 73%, Loss 0.563947]
Iter 130. [Val Acc 73%, Loss 0.566062]
Iter 140. [Val Acc 73%, Loss 0.562906]
Iter 150. [Val Acc 73%, Loss 0.566718]
Iter 160. [Val Acc 73%, Loss 0.566786]
Iter 170. [Val Acc 73%, Loss 0.564401]
Iter 180. [Val Acc 73%, Loss 0.568687]
Iter 190. [Val Acc 73%, Loss 0.565350]
Iter 200. [Val Acc 73%, Loss 0.564624]
Iter 210. [Val Acc 73%, Loss 0.564854]
Iter 220. [Val Acc 73%, Loss 0.565167]
Iter 230. [Val Acc 73%, Loss 0.563991]
Iter 240. [Val Acc 73%, Loss 0.564317]
Iter 250. [Val Acc 73%, Loss 0.567851]
Iter 260. [Val Acc 73%, Loss 0.562957]
Iter 270. [Val Acc 73%, Loss 0.564235]
Iter 280. [Val Acc 73%, Loss 0.563862]
Iter 290. [Val Acc 73%, Loss 0.561805]
Iter 300. [Val Acc 73%, Loss 0.562217]
Iter 310. [Val Acc 73%, Loss 0.566289]
Iter 320. [Val Acc 73%, Loss 0.563408]
Iter 330. [Val Acc 73%, Loss 0.563460]
Iter 340. [Val Acc 73%, Loss 0.565811]
Iter 350. [Val Acc 73%, Loss 0.564462]
Iter 360. [Val Acc 73%, Loss 0.562446]
Iter 370. [Val Acc 73%, Loss 0.561673]
Iter 380. [Val Acc 73%, Loss 0.562338]
Iter 390. [Val Acc 73%, Loss 0.562108]
Iter 400. [Val Acc 73%, Loss 0.563027]
Iter 410. [Val Acc 73%, Loss 0.561438]
Iter 420. [Val Acc 73%, Loss 0.562686]
Iter 430. [Val Acc 73%, Loss 0.561587]
Iter 440. [Val Acc 74%, Loss 0.559880]
Iter 450. [Val Acc 74%, Loss 0.560219]
Iter 460. [Val Acc 73%, Loss 0.562076]
Iter 470. [Val Acc 73%, Loss 0.560777]
Iter 480. [Val Acc 73%, Loss 0.564152]
Iter 490. [Val Acc 73%, Loss 0.560935]
Iter 500. [Val Acc 73%, Loss 0.561742]
Iter 510. [Val Acc 73%, Loss 0.561520]
Iter 520. [Val Acc 73%, Loss 0.559534]
Iter 530. [Val Acc 73%, Loss 0.560051]
Iter 540. [Val Acc 73%, Loss 0.560920]
Iter 550. [Val Acc 73%, Loss 0.562220]
Iter 560. [Val Acc 73%, Loss 0.559898]
Iter 570. [Val Acc 73%, Loss 0.560552]
Iter 580. [Val Acc 73%, Loss 0.563651]
Iter 590. [Val Acc 73%, Loss 0.562176]
Iter 600. [Val Acc 73%, Loss 0.560833]
Iter 610. [Val Acc 73%, Loss 0.562583]
Iter 620. [Val Acc 73%, Loss 0.563069]
Iter 630. [Val Acc 73%, Loss 0.560673]
Iter 640. [Val Acc 73%, Loss 0.561327]
Iter 650. [Val Acc 74%, Loss 0.560986]
Iter 660. [Val Acc 73%, Loss 0.560266]
Iter 670. [Val Acc 73%, Loss 0.562147]
Iter 680. [Val Acc 73%, Loss 0.562432]
Iter 690. [Val Acc 73%, Loss 0.560802]
Iter 700. [Val Acc 73%, Loss 0.559577]
Iter 710. [Val Acc 73%, Loss 0.559615]
Iter 720. [Val Acc 74%, Loss 0.559283]
Iter 730. [Val Acc 73%, Loss 0.559184]
Iter 740. [Val Acc 73%, Loss 0.561234]
Iter 750. [Val Acc 73%, Loss 0.561935]
Iter 760. [Val Acc 73%, Loss 0.560156]
Iter 770. [Val Acc 73%, Loss 0.561854]
Iter 780. [Val Acc 73%, Loss 0.560557]
Iter 790. [Val Acc 73%, Loss 0.560726]
Iter 800. [Val Acc 73%, Loss 0.561080]
Iter 810. [Val Acc 73%, Loss 0.560786]
Iter 820. [Val Acc 73%, Loss 0.567863]
Iter 830. [Val Acc 73%, Loss 0.561067]
Iter 840. [Val Acc 73%, Loss 0.560656]
Iter 850. [Val Acc 73%, Loss 0.562331]
Iter 860. [Val Acc 73%, Loss 0.562238]
Iter 870. [Val Acc 73%, Loss 0.562508]
Iter 880. [Val Acc 73%, Loss 0.559847]
Iter 890. [Val Acc 73%, Loss 0.559329]
Iter 900. [Val Acc 73%, Loss 0.559951]
Iter 910. [Val Acc 73%, Loss 0.559736]
Iter 920. [Val Acc 73%, Loss 0.559922]
Iter 930. [Val Acc 73%, Loss 0.560707]
Iter 940. [Val Acc 73%, Loss 0.565256]
Iter 950. [Val Acc 73%, Loss 0.562449]
Iter 960. [Val Acc 73%, Loss 0.560893]
Iter 970. [Val Acc 73%, Loss 0.562585]
Iter 980. [Val Acc 73%, Loss 0.560763]
Iter 990. [Val Acc 73%, Loss 0.560838]
Iter 1000. [Val Acc 73%, Loss 0.561143]
-------------------------
monte_carlo: 1
mu= 0.1 batch_size= 1 :
Iter 10. [Val Acc 50%, Loss 3.606403]
Iter 20. [Val Acc 50%, Loss 3.232711]
Iter 30. [Val Acc 50%, Loss 2.954263]
Iter 40. [Val Acc 51%, Loss 2.713782]
Iter 50. [Val Acc 51%, Loss 2.505076]
Iter 60. [Val Acc 52%, Loss 2.304634]
Iter 70. [Val Acc 53%, Loss 2.157649]
Iter 80. [Val Acc 53%, Loss 2.025197]
Iter 90. [Val Acc 54%, Loss 1.911352]
Iter 100. [Val Acc 54%, Loss 1.805560]
Iter 110. [Val Acc 55%, Loss 1.721055]
Iter 120. [Val Acc 55%, Loss 1.644382]
Iter 130. [Val Acc 56%, Loss 1.572820]
Iter 140. [Val Acc 56%, Loss 1.499186]
Iter 150. [Val Acc 57%, Loss 1.433355]
Iter 160. [Val Acc 57%, Loss 1.375669]
Iter 170. [Val Acc 58%, Loss 1.329589]
Iter 180. [Val Acc 58%, Loss 1.271582]
Iter 190. [Val Acc 58%, Loss 1.218902]
Iter 200. [Val Acc 59%, Loss 1.163627]
Iter 210. [Val Acc 60%, Loss 1.125665]
Iter 220. [Val Acc 60%, Loss 1.092377]
Iter 230. [Val Acc 61%, Loss 1.057642]
Iter 240. [Val Acc 61%, Loss 1.023549]
Iter 250. [Val Acc 61%, Loss 0.991729]
Iter 260. [Val Acc 62%, Loss 0.965199]
Iter 270. [Val Acc 62%, Loss 0.934834]
Iter 280. [Val Acc 63%, Loss 0.912913]
Iter 290. [Val Acc 63%, Loss 0.887457]
Iter 300. [Val Acc 63%, Loss 0.869442]
Iter 310. [Val Acc 64%, Loss 0.847111]
Iter 320. [Val Acc 64%, Loss 0.823289]
Iter 330. [Val Acc 64%, Loss 0.810874]
Iter 340. [Val Acc 64%, Loss 0.798560]
Iter 350. [Val Acc 65%, Loss 0.780805]
Iter 360. [Val Acc 65%, Loss 0.766289]
Iter 370. [Val Acc 66%, Loss 0.754752]
Iter 380. [Val Acc 66%, Loss 0.743504]
Iter 390. [Val Acc 66%, Loss 0.730939]
Iter 400. [Val Acc 66%, Loss 0.723672]
Iter 410. [Val Acc 66%, Loss 0.717218]
Iter 420. [Val Acc 67%, Loss 0.715492]
Iter 430. [Val Acc 67%, Loss 0.695718]
Iter 440. [Val Acc 67%, Loss 0.689669]
Iter 450. [Val Acc 67%, Loss 0.684444]
Iter 460. [Val Acc 68%, Loss 0.675668]
Iter 470. [Val Acc 68%, Loss 0.673336]
Iter 480. [Val Acc 68%, Loss 0.666558]
Iter 490. [Val Acc 68%, Loss 0.660774]
Iter 500. [Val Acc 68%, Loss 0.654241]
Iter 510. [Val Acc 69%, Loss 0.649946]
Iter 520. [Val Acc 69%, Loss 0.646229]
Iter 530. [Val Acc 69%, Loss 0.642368]
Iter 540. [Val Acc 69%, Loss 0.640598]
Iter 550. [Val Acc 69%, Loss 0.635907]
Iter 560. [Val Acc 69%, Loss 0.633418]
Iter 570. [Val Acc 70%, Loss 0.628302]
Iter 580. [Val Acc 70%, Loss 0.627600]
Iter 590. [Val Acc 70%, Loss 0.626729]
Iter 600. [Val Acc 70%, Loss 0.621411]
Iter 610. [Val Acc 70%, Loss 0.621846]
Iter 620. [Val Acc 70%, Loss 0.617425]
Iter 630. [Val Acc 70%, Loss 0.613802]
Iter 640. [Val Acc 70%, Loss 0.614414]
Iter 650. [Val Acc 70%, Loss 0.613460]
Iter 660. [Val Acc 70%, Loss 0.608775]
Iter 670. [Val Acc 70%, Loss 0.606546]
Iter 680. [Val Acc 71%, Loss 0.603055]
Iter 690. [Val Acc 71%, Loss 0.601859]
Iter 700. [Val Acc 71%, Loss 0.601248]
Iter 710. [Val Acc 71%, Loss 0.599102]
Iter 720. [Val Acc 71%, Loss 0.597411]
Iter 730. [Val Acc 71%, Loss 0.594596]
Iter 740. [Val Acc 71%, Loss 0.592610]
Iter 750. [Val Acc 71%, Loss 0.592885]
Iter 760. [Val Acc 71%, Loss 0.592737]
Iter 770. [Val Acc 71%, Loss 0.594528]
Iter 780. [Val Acc 72%, Loss 0.590904]
Iter 790. [Val Acc 72%, Loss 0.592990]
Iter 800. [Val Acc 72%, Loss 0.587860]
Iter 810. [Val Acc 72%, Loss 0.586981]
Iter 820. [Val Acc 72%, Loss 0.585866]
Iter 830. [Val Acc 72%, Loss 0.588084]
Iter 840. [Val Acc 72%, Loss 0.585397]
Iter 850. [Val Acc 72%, Loss 0.583619]
Iter 860. [Val Acc 72%, Loss 0.582261]
Iter 870. [Val Acc 72%, Loss 0.580161]
Iter 880. [Val Acc 72%, Loss 0.580038]
Iter 890. [Val Acc 72%, Loss 0.578393]
Iter 900. [Val Acc 72%, Loss 0.577898]
Iter 910. [Val Acc 72%, Loss 0.579943]
Iter 920. [Val Acc 72%, Loss 0.577038]
Iter 930. [Val Acc 73%, Loss 0.575450]
Iter 940. [Val Acc 72%, Loss 0.579716]
Iter 950. [Val Acc 72%, Loss 0.574575]
Iter 960. [Val Acc 72%, Loss 0.575289]
Iter 970. [Val Acc 72%, Loss 0.576867]
Iter 980. [Val Acc 72%, Loss 0.574316]
Iter 990. [Val Acc 73%, Loss 0.573036]
Iter 1000. [Val Acc 72%, Loss 0.572772]
-------------------------
mu= 0.1 batch_size= 50000 :
Iter 10. [Val Acc 60%, Loss 0.736115]
Iter 20. [Val Acc 65%, Loss 0.686533]
Iter 30. [Val Acc 68%, Loss 0.652574]
Iter 40. [Val Acc 69%, Loss 0.627356]
Iter 50. [Val Acc 70%, Loss 0.610556]
Iter 60. [Val Acc 71%, Loss 0.603844]
Iter 70. [Val Acc 72%, Loss 0.592599]
Iter 80. [Val Acc 72%, Loss 0.588085]
Iter 90. [Val Acc 72%, Loss 0.584503]
Iter 100. [Val Acc 72%, Loss 0.582740]
Iter 110. [Val Acc 72%, Loss 0.577937]
Iter 120. [Val Acc 72%, Loss 0.576994]
Iter 130. [Val Acc 73%, Loss 0.573602]
Iter 140. [Val Acc 73%, Loss 0.571907]
Iter 150. [Val Acc 73%, Loss 0.571011]
Iter 160. [Val Acc 73%, Loss 0.569678]
Iter 170. [Val Acc 73%, Loss 0.570306]
Iter 180. [Val Acc 73%, Loss 0.570271]
Iter 190. [Val Acc 73%, Loss 0.568046]
Iter 200. [Val Acc 73%, Loss 0.570174]
Iter 210. [Val Acc 73%, Loss 0.568895]
Iter 220. [Val Acc 73%, Loss 0.569121]
Iter 230. [Val Acc 73%, Loss 0.568084]
Iter 240. [Val Acc 73%, Loss 0.567097]
Iter 250. [Val Acc 73%, Loss 0.567504]
Iter 260. [Val Acc 73%, Loss 0.567608]
Iter 270. [Val Acc 73%, Loss 0.567378]
Iter 280. [Val Acc 73%, Loss 0.568777]
Iter 290. [Val Acc 73%, Loss 0.566003]
Iter 300. [Val Acc 73%, Loss 0.567384]
Iter 310. [Val Acc 73%, Loss 0.568386]
Iter 320. [Val Acc 73%, Loss 0.564794]
Iter 330. [Val Acc 73%, Loss 0.566777]
Iter 340. [Val Acc 73%, Loss 0.564675]
Iter 350. [Val Acc 73%, Loss 0.564413]
Iter 360. [Val Acc 73%, Loss 0.568015]
Iter 370. [Val Acc 73%, Loss 0.565861]
Iter 380. [Val Acc 73%, Loss 0.564881]
Iter 390. [Val Acc 73%, Loss 0.564935]
Iter 400. [Val Acc 73%, Loss 0.564168]
Iter 410. [Val Acc 73%, Loss 0.563537]
Iter 420. [Val Acc 73%, Loss 0.564986]
Iter 430. [Val Acc 73%, Loss 0.564259]
Iter 440. [Val Acc 73%, Loss 0.564824]
Iter 450. [Val Acc 73%, Loss 0.565714]
Iter 460. [Val Acc 73%, Loss 0.565908]
Iter 470. [Val Acc 73%, Loss 0.563462]
Iter 480. [Val Acc 73%, Loss 0.563564]
Iter 490. [Val Acc 73%, Loss 0.563662]
Iter 500. [Val Acc 73%, Loss 0.564205]
Iter 510. [Val Acc 73%, Loss 0.563514]
Iter 520. [Val Acc 73%, Loss 0.564427]
Iter 530. [Val Acc 73%, Loss 0.565638]
Iter 540. [Val Acc 73%, Loss 0.563486]
Iter 550. [Val Acc 73%, Loss 0.563062]
Iter 560. [Val Acc 73%, Loss 0.564175]
Iter 570. [Val Acc 73%, Loss 0.563936]
Iter 580. [Val Acc 73%, Loss 0.564311]
Iter 590. [Val Acc 73%, Loss 0.562764]
Iter 600. [Val Acc 73%, Loss 0.562597]
Iter 610. [Val Acc 73%, Loss 0.565040]
Iter 620. [Val Acc 73%, Loss 0.565235]
Iter 630. [Val Acc 73%, Loss 0.562720]
Iter 640. [Val Acc 73%, Loss 0.563254]
Iter 650. [Val Acc 73%, Loss 0.563242]
Iter 660. [Val Acc 73%, Loss 0.563620]
Iter 670. [Val Acc 73%, Loss 0.564230]
Iter 680. [Val Acc 73%, Loss 0.564952]
Iter 690. [Val Acc 73%, Loss 0.566127]
Iter 700. [Val Acc 73%, Loss 0.562753]
Iter 710. [Val Acc 73%, Loss 0.563075]
Iter 720. [Val Acc 73%, Loss 0.563789]
Iter 730. [Val Acc 73%, Loss 0.562714]
Iter 740. [Val Acc 73%, Loss 0.561236]
Iter 750. [Val Acc 73%, Loss 0.561560]
Iter 760. [Val Acc 73%, Loss 0.561698]
Iter 770. [Val Acc 73%, Loss 0.562777]
Iter 780. [Val Acc 73%, Loss 0.561599]
Iter 790. [Val Acc 73%, Loss 0.563078]
Iter 800. [Val Acc 73%, Loss 0.564514]
Iter 810. [Val Acc 73%, Loss 0.567656]
Iter 820. [Val Acc 73%, Loss 0.563739]
Iter 830. [Val Acc 73%, Loss 0.563306]
Iter 840. [Val Acc 73%, Loss 0.564770]
Iter 850. [Val Acc 73%, Loss 0.564090]
Iter 860. [Val Acc 73%, Loss 0.560671]
Iter 870. [Val Acc 73%, Loss 0.559944]
Iter 880. [Val Acc 73%, Loss 0.561137]
Iter 890. [Val Acc 73%, Loss 0.561869]
Iter 900. [Val Acc 74%, Loss 0.560795]
Iter 910. [Val Acc 73%, Loss 0.562739]
Iter 920. [Val Acc 73%, Loss 0.560706]
Iter 930. [Val Acc 73%, Loss 0.559462]
Iter 940. [Val Acc 73%, Loss 0.561349]
Iter 950. [Val Acc 73%, Loss 0.562603]
Iter 960. [Val Acc 73%, Loss 0.563293]
Iter 970. [Val Acc 73%, Loss 0.562432]
Iter 980. [Val Acc 73%, Loss 0.562910]
Iter 990. [Val Acc 73%, Loss 0.562498]
Iter 1000. [Val Acc 73%, Loss 0.561585]
-------------------------
monte_carlo: 2
mu= 0.1 batch_size= 1 :
Iter 10. [Val Acc 49%, Loss 4.088643]
Iter 20. [Val Acc 49%, Loss 3.643840]
Iter 30. [Val Acc 49%, Loss 3.286488]
Iter 40. [Val Acc 49%, Loss 2.997371]
Iter 50. [Val Acc 50%, Loss 2.791540]
Iter 60. [Val Acc 50%, Loss 2.586161]
Iter 70. [Val Acc 51%, Loss 2.387459]
Iter 80. [Val Acc 51%, Loss 2.258004]
Iter 90. [Val Acc 52%, Loss 2.107343]
Iter 100. [Val Acc 53%, Loss 1.984522]
Iter 110. [Val Acc 53%, Loss 1.879489]
Iter 120. [Val Acc 54%, Loss 1.769485]
Iter 130. [Val Acc 54%, Loss 1.682595]
Iter 140. [Val Acc 55%, Loss 1.587613]
Iter 150. [Val Acc 56%, Loss 1.518055]
Iter 160. [Val Acc 57%, Loss 1.462195]
Iter 170. [Val Acc 57%, Loss 1.389192]
Iter 180. [Val Acc 58%, Loss 1.337626]
Iter 190. [Val Acc 59%, Loss 1.273107]
Iter 200. [Val Acc 59%, Loss 1.222966]
Iter 210. [Val Acc 60%, Loss 1.176070]
Iter 220. [Val Acc 61%, Loss 1.142139]
Iter 230. [Val Acc 61%, Loss 1.100943]
Iter 240. [Val Acc 61%, Loss 1.062682]
Iter 250. [Val Acc 62%, Loss 1.029151]
Iter 260. [Val Acc 62%, Loss 1.002743]
Iter 270. [Val Acc 63%, Loss 0.980588]
Iter 280. [Val Acc 63%, Loss 0.950483]
Iter 290. [Val Acc 63%, Loss 0.925953]
Iter 300. [Val Acc 64%, Loss 0.905360]
Iter 310. [Val Acc 64%, Loss 0.885543]
Iter 320. [Val Acc 65%, Loss 0.861586]
Iter 330. [Val Acc 65%, Loss 0.842996]
Iter 340. [Val Acc 65%, Loss 0.830159]
Iter 350. [Val Acc 65%, Loss 0.805328]
Iter 360. [Val Acc 66%, Loss 0.794037]
Iter 370. [Val Acc 66%, Loss 0.780222]
Iter 380. [Val Acc 66%, Loss 0.768399]
Iter 390. [Val Acc 66%, Loss 0.752948]
Iter 400. [Val Acc 67%, Loss 0.740897]
Iter 410. [Val Acc 67%, Loss 0.728946]
Iter 420. [Val Acc 67%, Loss 0.723727]
Iter 430. [Val Acc 67%, Loss 0.710282]
Iter 440. [Val Acc 68%, Loss 0.702487]
Iter 450. [Val Acc 68%, Loss 0.694278]
Iter 460. [Val Acc 68%, Loss 0.688999]
Iter 470. [Val Acc 69%, Loss 0.680275]
Iter 480. [Val Acc 69%, Loss 0.675113]
Iter 490. [Val Acc 69%, Loss 0.674970]
Iter 500. [Val Acc 69%, Loss 0.668249]
Iter 510. [Val Acc 69%, Loss 0.658802]
Iter 520. [Val Acc 69%, Loss 0.652548]
Iter 530. [Val Acc 69%, Loss 0.651298]
Iter 540. [Val Acc 70%, Loss 0.645338]
Iter 550. [Val Acc 70%, Loss 0.640313]
Iter 560. [Val Acc 70%, Loss 0.636863]
Iter 570. [Val Acc 70%, Loss 0.630801]
Iter 580. [Val Acc 70%, Loss 0.626878]
Iter 590. [Val Acc 70%, Loss 0.622515]
Iter 600. [Val Acc 71%, Loss 0.618195]
Iter 610. [Val Acc 70%, Loss 0.618347]
Iter 620. [Val Acc 71%, Loss 0.612297]
Iter 630. [Val Acc 71%, Loss 0.610601]
Iter 640. [Val Acc 71%, Loss 0.606956]
Iter 650. [Val Acc 71%, Loss 0.604283]
Iter 660. [Val Acc 71%, Loss 0.602555]
Iter 670. [Val Acc 71%, Loss 0.602099]
Iter 680. [Val Acc 71%, Loss 0.598500]
Iter 690. [Val Acc 71%, Loss 0.599564]
Iter 700. [Val Acc 72%, Loss 0.596450]
Iter 710. [Val Acc 72%, Loss 0.595301]
Iter 720. [Val Acc 71%, Loss 0.592705]
Iter 730. [Val Acc 72%, Loss 0.591639]
Iter 740. [Val Acc 72%, Loss 0.588997]
Iter 750. [Val Acc 72%, Loss 0.588292]
Iter 760. [Val Acc 72%, Loss 0.586960]
Iter 770. [Val Acc 72%, Loss 0.586938]
Iter 780. [Val Acc 72%, Loss 0.585840]
Iter 790. [Val Acc 72%, Loss 0.583550]
Iter 800. [Val Acc 72%, Loss 0.581137]
Iter 810. [Val Acc 72%, Loss 0.581396]
Iter 820. [Val Acc 72%, Loss 0.580694]
Iter 830. [Val Acc 72%, Loss 0.579579]
Iter 840. [Val Acc 72%, Loss 0.578937]
Iter 850. [Val Acc 72%, Loss 0.578088]
Iter 860. [Val Acc 72%, Loss 0.576692]
Iter 870. [Val Acc 72%, Loss 0.575402]
Iter 880. [Val Acc 72%, Loss 0.575965]
Iter 890. [Val Acc 73%, Loss 0.573956]
Iter 900. [Val Acc 72%, Loss 0.573690]
Iter 910. [Val Acc 72%, Loss 0.574301]
Iter 920. [Val Acc 73%, Loss 0.573690]
Iter 930. [Val Acc 72%, Loss 0.573711]
Iter 940. [Val Acc 73%, Loss 0.572138]
Iter 950. [Val Acc 72%, Loss 0.573484]
Iter 960. [Val Acc 73%, Loss 0.571933]
Iter 970. [Val Acc 73%, Loss 0.572595]
Iter 980. [Val Acc 72%, Loss 0.576842]
Iter 990. [Val Acc 73%, Loss 0.571624]
Iter 1000. [Val Acc 73%, Loss 0.572718]
-------------------------
mu= 0.1 batch_size= 50000 :
Iter 10. [Val Acc 72%, Loss 0.588926]
Iter 20. [Val Acc 72%, Loss 0.582369]
Iter 30. [Val Acc 73%, Loss 0.577977]
Iter 40. [Val Acc 73%, Loss 0.575667]
Iter 50. [Val Acc 73%, Loss 0.574534]
Iter 60. [Val Acc 73%, Loss 0.571842]
Iter 70. [Val Acc 73%, Loss 0.571034]
Iter 80. [Val Acc 73%, Loss 0.568841]
Iter 90. [Val Acc 73%, Loss 0.568537]
Iter 100. [Val Acc 73%, Loss 0.568093]
Iter 110. [Val Acc 73%, Loss 0.566364]
Iter 120. [Val Acc 73%, Loss 0.568557]
Iter 130. [Val Acc 73%, Loss 0.568008]
Iter 140. [Val Acc 73%, Loss 0.568118]
Iter 150. [Val Acc 73%, Loss 0.567871]
Iter 160. [Val Acc 73%, Loss 0.567331]
Iter 170. [Val Acc 73%, Loss 0.567699]
Iter 180. [Val Acc 73%, Loss 0.565132]
Iter 190. [Val Acc 73%, Loss 0.567102]
Iter 200. [Val Acc 73%, Loss 0.567063]
Iter 210. [Val Acc 73%, Loss 0.566032]
Iter 220. [Val Acc 73%, Loss 0.564880]
Iter 230. [Val Acc 73%, Loss 0.566003]
Iter 240. [Val Acc 73%, Loss 0.563302]
Iter 250. [Val Acc 73%, Loss 0.564889]
Iter 260. [Val Acc 73%, Loss 0.563869]
Iter 270. [Val Acc 73%, Loss 0.564403]
Iter 280. [Val Acc 73%, Loss 0.565197]
Iter 290. [Val Acc 73%, Loss 0.565225]
Iter 300. [Val Acc 73%, Loss 0.563910]
Iter 310. [Val Acc 73%, Loss 0.562909]
Iter 320. [Val Acc 73%, Loss 0.563186]
Iter 330. [Val Acc 73%, Loss 0.563382]
Iter 340. [Val Acc 73%, Loss 0.563543]
Iter 350. [Val Acc 73%, Loss 0.564477]
Iter 360. [Val Acc 73%, Loss 0.563013]
Iter 370. [Val Acc 73%, Loss 0.562193]
Iter 380. [Val Acc 73%, Loss 0.561940]
Iter 390. [Val Acc 73%, Loss 0.561862]
Iter 400. [Val Acc 73%, Loss 0.562086]
Iter 410. [Val Acc 73%, Loss 0.563611]
Iter 420. [Val Acc 73%, Loss 0.562986]
Iter 430. [Val Acc 73%, Loss 0.563011]
Iter 440. [Val Acc 73%, Loss 0.562432]
Iter 450. [Val Acc 73%, Loss 0.563435]
Iter 460. [Val Acc 73%, Loss 0.563261]
Iter 470. [Val Acc 73%, Loss 0.564406]
Iter 480. [Val Acc 73%, Loss 0.562204]
Iter 490. [Val Acc 73%, Loss 0.562455]
Iter 500. [Val Acc 73%, Loss 0.562971]
Iter 510. [Val Acc 73%, Loss 0.563270]
Iter 520. [Val Acc 73%, Loss 0.562334]
Iter 530. [Val Acc 73%, Loss 0.561062]
Iter 540. [Val Acc 73%, Loss 0.562285]
Iter 550. [Val Acc 73%, Loss 0.561582]
Iter 560. [Val Acc 73%, Loss 0.559796]
Iter 570. [Val Acc 73%, Loss 0.561570]
Iter 580. [Val Acc 74%, Loss 0.562317]
Iter 590. [Val Acc 73%, Loss 0.561832]
Iter 600. [Val Acc 73%, Loss 0.560521]
Iter 610. [Val Acc 73%, Loss 0.560394]
Iter 620. [Val Acc 73%, Loss 0.561316]
Iter 630. [Val Acc 74%, Loss 0.560536]
Iter 640. [Val Acc 73%, Loss 0.560485]
Iter 650. [Val Acc 73%, Loss 0.560967]
Iter 660. [Val Acc 74%, Loss 0.562116]
Iter 670. [Val Acc 73%, Loss 0.559665]
Iter 680. [Val Acc 73%, Loss 0.559768]
Iter 690. [Val Acc 73%, Loss 0.560675]
Iter 700. [Val Acc 73%, Loss 0.561690]
Iter 710. [Val Acc 73%, Loss 0.564173]
Iter 720. [Val Acc 73%, Loss 0.560785]
Iter 730. [Val Acc 73%, Loss 0.563405]
Iter 740. [Val Acc 73%, Loss 0.565020]
Iter 750. [Val Acc 73%, Loss 0.561862]
Iter 760. [Val Acc 74%, Loss 0.561143]
Iter 770. [Val Acc 73%, Loss 0.562776]
Iter 780. [Val Acc 73%, Loss 0.563701]
Iter 790. [Val Acc 73%, Loss 0.560083]
Iter 800. [Val Acc 73%, Loss 0.560464]
Iter 810. [Val Acc 73%, Loss 0.560005]
Iter 820. [Val Acc 73%, Loss 0.561094]
Iter 830. [Val Acc 73%, Loss 0.560373]
Iter 840. [Val Acc 73%, Loss 0.561206]
Iter 850. [Val Acc 73%, Loss 0.565063]
Iter 860. [Val Acc 73%, Loss 0.562654]
Iter 870. [Val Acc 73%, Loss 0.561572]
Iter 880. [Val Acc 74%, Loss 0.563063]
Iter 890. [Val Acc 73%, Loss 0.564624]
Iter 900. [Val Acc 73%, Loss 0.562960]
Iter 910. [Val Acc 73%, Loss 0.561574]
Iter 920. [Val Acc 74%, Loss 0.561824]
Iter 930. [Val Acc 73%, Loss 0.561352]
Iter 940. [Val Acc 73%, Loss 0.563464]
Iter 950. [Val Acc 73%, Loss 0.562971]
Iter 960. [Val Acc 73%, Loss 0.561951]
Iter 970. [Val Acc 74%, Loss 0.562226]
Iter 980. [Val Acc 73%, Loss 0.560642]
Iter 990. [Val Acc 74%, Loss 0.559579]
Iter 1000. [Val Acc 73%, Loss 0.562161]
-------------------------
```

```
[ ]: Text(0.5, 1.0, 'Batch size convergance properties')
```



For $\mu = 0.1$ and batch size 1 we can see from this figure that it takes more iterations for the validation loss to converge (compared with batch size 50000) because the estimation of the gradient is based on one sample, which makes this estimation very noisy. Through the training procedure we see that a minibatch with smaller size suffer from higher variance when applying Monte Carlo experiment with random parameters initialization.

2.0.6 Part (h) -- 15%

Using the values of `w` and `b` from part (g), compute your training accuracy, validation accuracy, and test accuracy. Are there any differences between those three values? If so, why?

```
[ ]: train_ys=pred(opt_w,opt_b,train_norm_xs)
test_ys=pred(opt_w,opt_b,test_norm_xs)
val_ys=pred(opt_w,opt_b,val_norm_xs)



train_acc = get_accuracy(train_ys,train_ts)
val_acc = get_accuracy(val_ys,val_ts)
test_acc =  get_accuracy(test_ys,test_ts)

print('train_acc = ', train_acc, ' val_acc = ', val_acc, ' test_acc = ',␣
 ↪test_acc)
```

```
train_acc =  0.7315905877234328  val_acc =  0.73532  test_acc =
0.7247143133836916
```

Both train and validation accuracies are better than the test accuracy, we explain this phenomenoa by the fact that we didn't see the test data during the model building process, so we expect to encounter minor degradation in the model performance over the new unseen data.

Moreover, we can see that the gap between the train and test data is not significant and therefore we conclude that the model can extract meaningful information from the training dataset. Meaning that the model is capable to perform similarly over unseen data, e.g., the model is well generalized.

Regarding the validation accuracy, we have tuned our model hyperparameters(Learning rate and Batch size) such that it achieves the best performance over the validation data, therefore, we shall expect to achieve better results in the validation data than the test data, and this is indeed the result that we've achieved.

2.0.7 Part (i) -- 15%

Writing a classifier like this is instructive, and helps you understand what happens when we train a model. However, in practice, we rarely write model building and training code from scratch. Instead, we typically use one of the well-tested libraries available in a package.

Use `sklearn.linear_model.LogisticRegression` to build a linear classifier, and make predictions about the test set. Start by reading the API documentation here.

Compute the training, validation and test accuracy of this model.

```
[ ]: import sklearn.linear_model

model = sklearn.linear_model.LogisticRegression()
model.fit(train_norm_xs, train_ts.squeeze())
train_ys = model.predict(train_norm_xs)
test_ys = model.predict(test_norm_xs)
val_ys = model.predict(val_norm_xs)
train_acc = get_accuracy(train_ys,train_ts)
val_acc = get_accuracy(val_ys,val_ts)
```

```
test_acc =  get_accuracy(test_ys,test_ts)

print('train_acc = ', train_acc, ' val_acc = ', val_acc, ' test_acc = ',␣
 ↪test_acc)
```

```
train_acc =  0.7323979067715698  val_acc =  0.73642  test_acc =
0.7265736974627155
```

This parts helps by checking if the code worked. Check if you get similar results, if not repair your code