12. Training neural networks for classification problems and project proposal discussion

M.A.Z. Chowdhury and M.A. Oehlschlaeger

Department of Mechanical, Aerospace and Nuclear Engineering
Rensselaer Polytechnic Institute
Troy, New York

chowdm@rpi.edu
oehlsm@rpi.edu

MANE 4962 and 6962

Regular announcement

- ☞ Quiz 4 on Feb 27. ✓
- ☞ Quiz 4 is based on notes from Lecture 10.
- ☞ HW 4 is due Feb 27. ✓ → March 2
- ☞ Initial project proposal due March 2. ✓

Outline

Layers
Units per layer $\boxed{x_1, x_2}$ $\log x_1, \log x_2$
'relu' learning Rate = 0.001 $x_2, \log x_1$

- We solved regression problems using a fully connected neural network, $y \in \mathbb{R}$ ← $y_1 \in \mathbb{R}$, $y_2 \in \mathbb{R}$
- Solve a binary classification problem using a fully connected neural network $y \in [0, 1]$
- Solve a multi-class classification problem using a fully connected neural network~~s~~ $y \in \mathbb{R}^K$, K is the number of classes
- HW4 Hints
- Discussion about the initial and revised project proposal submission
- Study the notebooks for today's class

$$\sigma(z) = \frac{1}{1+e^{-z}} \in [0,1]$$

$\lim_{\to -\infty}$ $\lim_{\to \infty}$

decision

< 0.5 0
≥ 0.5 1

MNIST Digits

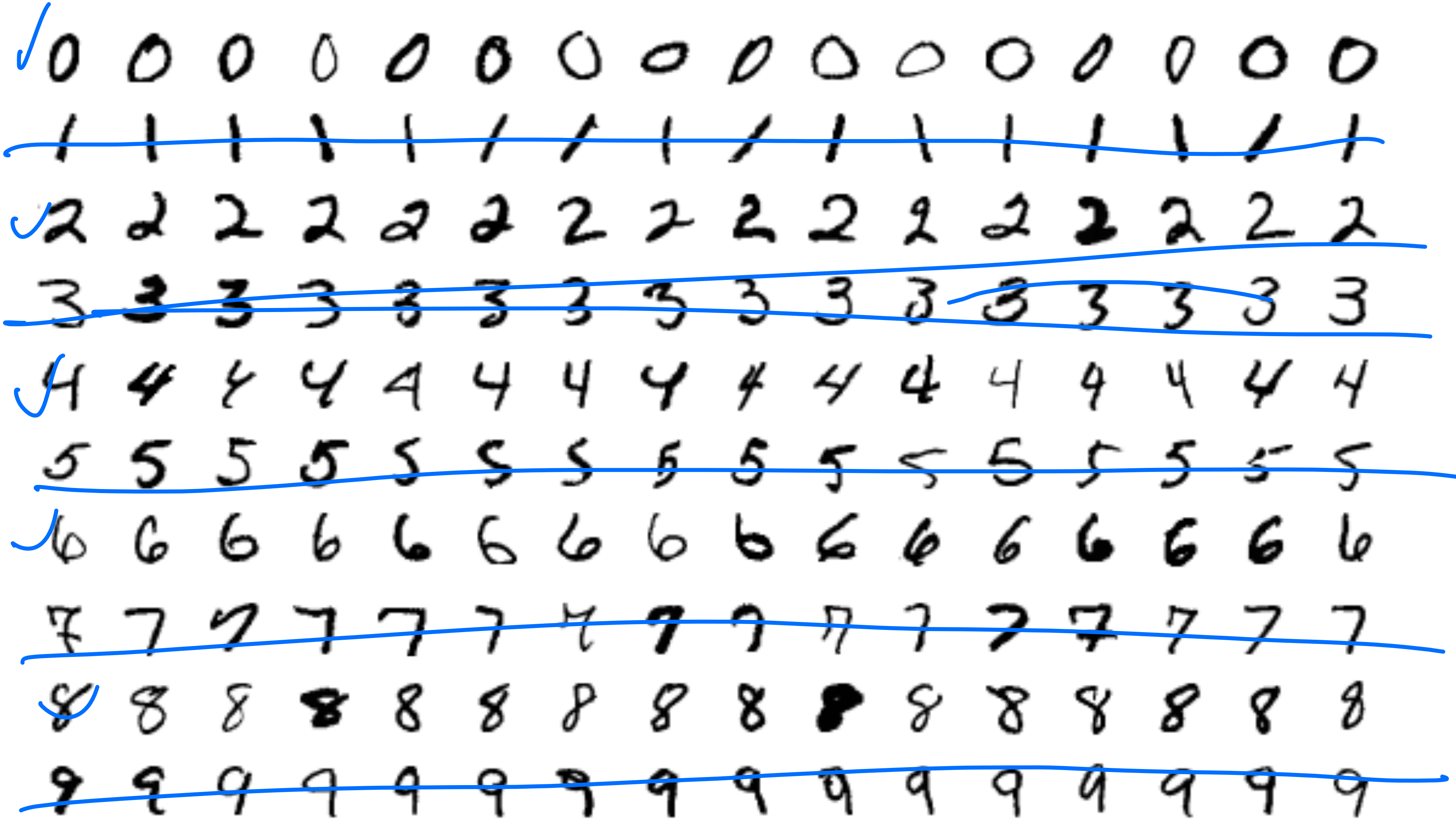
→ SGD → NN

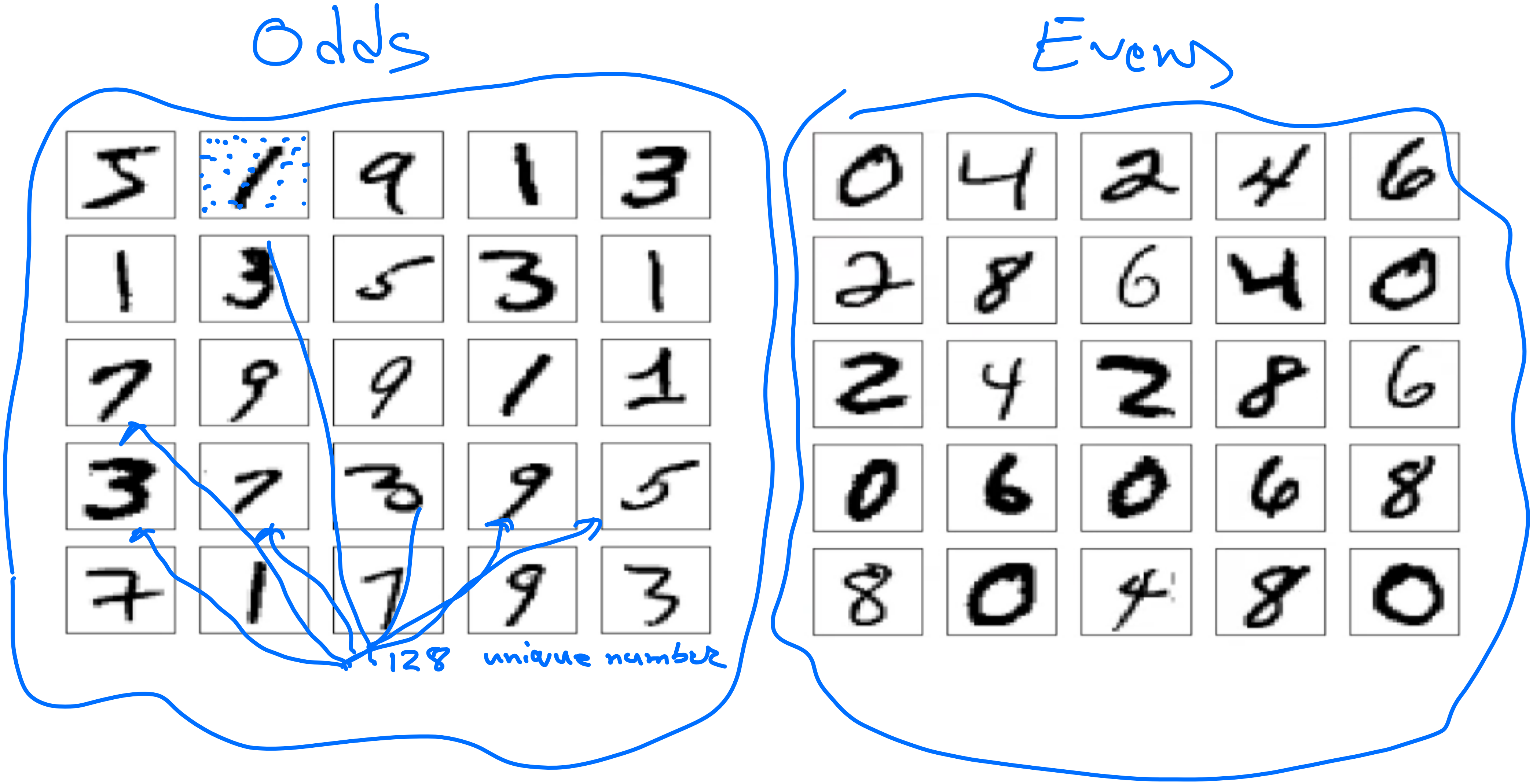
batch size = 10

mini-batch stochastic gradient descent

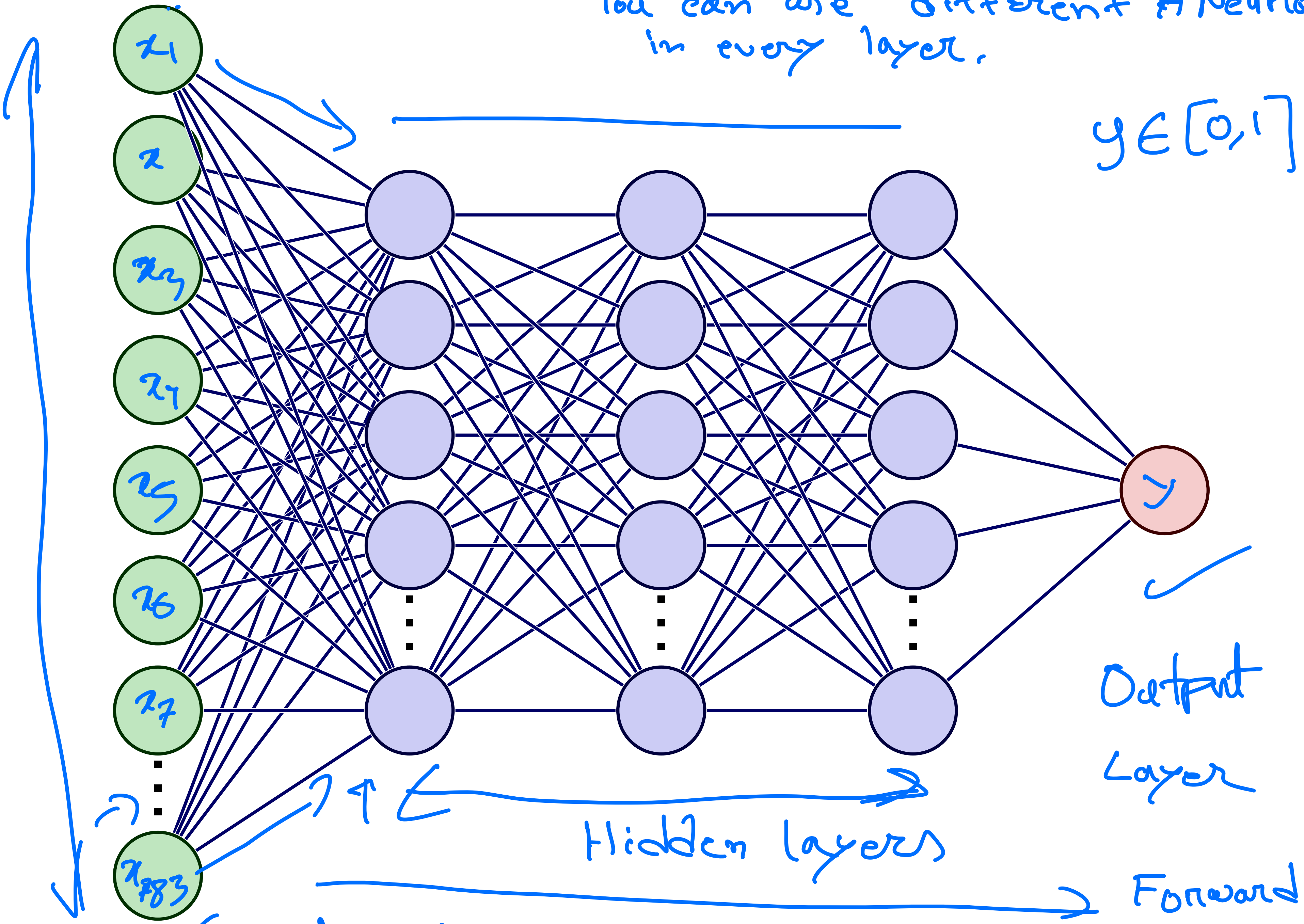
```
from keras.datasets import mnist

(x_train, y_train), (x_test, y_test) = mnist.load_data()

print("training images shapes: ", x_train.shape)
print("testing images shapes: ", x_test.shape)
print("training targets shapes: ", y_train.shape)
print("testing targers shapes: ", y_test.shape)

plt.imshow(x_train[0], cmap=plt.cm.gray_r, interpolation="nearest")
print(x_train[0])
```

Visualization of MNIST digits




Odds
Evens
128 unique number

You can use different # Neurons in every layer.



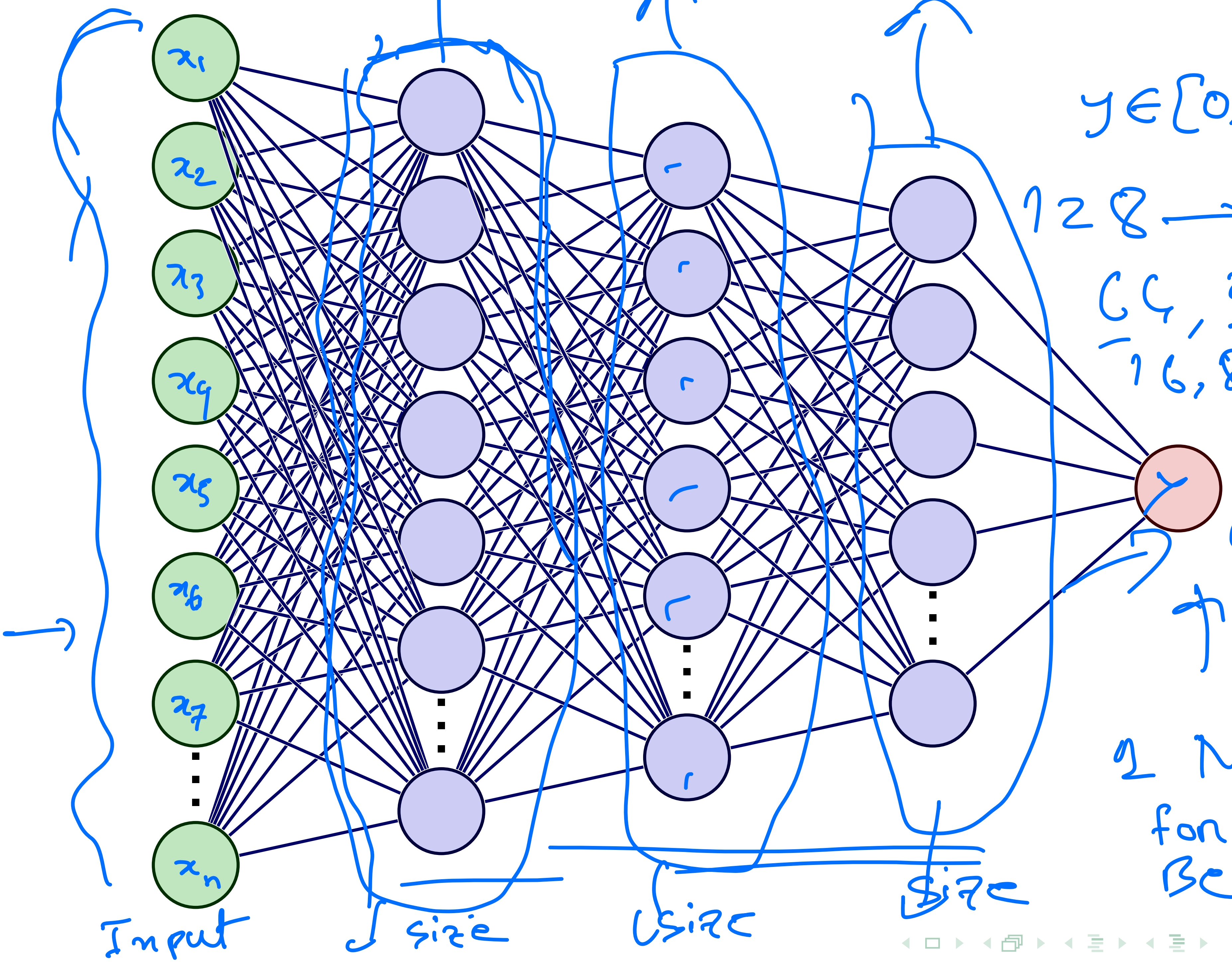
$y \in [0,1]$

Output Layer

Hidden layers

Forward

Input layer



$y \in [0, 1]$

128 → 1

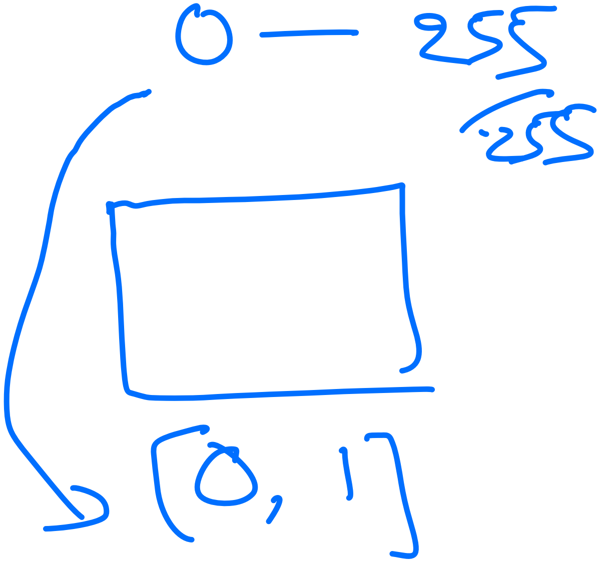
64, 32, 16, 8, 4, 2,

0, 1

1 Neuron for Bc

Implementation in tensorflow

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
import matplotlib.pyplot as plt

(x_train, y_train), (x_test, y_test) = mnist.load_data()

x_train = x_train.reshape(60000, 784)
x_test = x_test.reshape(10000, 784)

x_train = x_train.astype('float32') / 255
x_test = x_test.astype('float32') / 255
y_train = y_train % 2
y_test = y_test % 2
```



Normalizing

↳ tags the images

0 % 2 → 0

1 % 2 → 1

3 % 2 → 1

5 % → 1

0 — 2

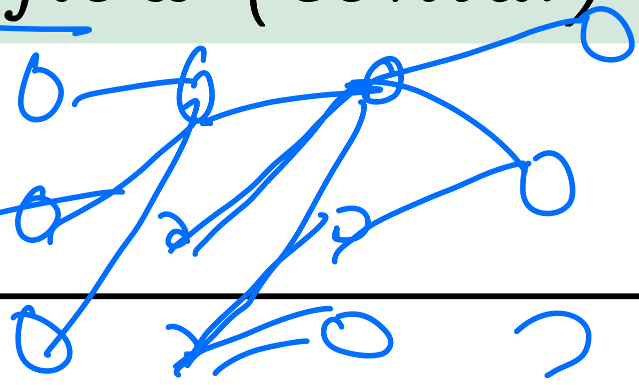
0.5 — 1.7

Step 1: Preprocess the data as required by the problem.

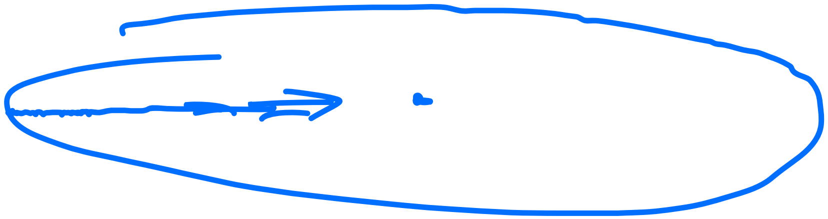
Implementation in tensorflow (contd.)

ReLU → Stickler

Sigmoid → Allow -ve features to pass through



```
model = Sequential()
model.add(Dense(512, activation='relu', input_shape=(784,)))
model.add(Dense(256, activation='relu'))
model.add(Dense(128, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])
model.summary()
```



CV →

Hyperparameter

Step 2: Define model. You may need to perform cross-validation to find the optimal architecture.

→ Sigmoid → estimated probability

Implementation in tensorflow (contd.)

tanh → activation , Goodfellow → 90% ReLU

```
# Train the model
history = model.fit(x_train, y_train, epochs=10,
batch_size=64, validation_split=0.3)

plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')

plt.show()
```

loss

0 1

$x = 0 \quad , = 0.5 = y$

$se = (-0.5)^2$

$= 0.25$

$= -[\cancel{0 \ln 0.5} + (1-0) \ln(-0.5)]$

$= \ln(-0.5)$

Step 3: Train model and analyze further. You can add more steps of postprocessing as needed.

Validation data

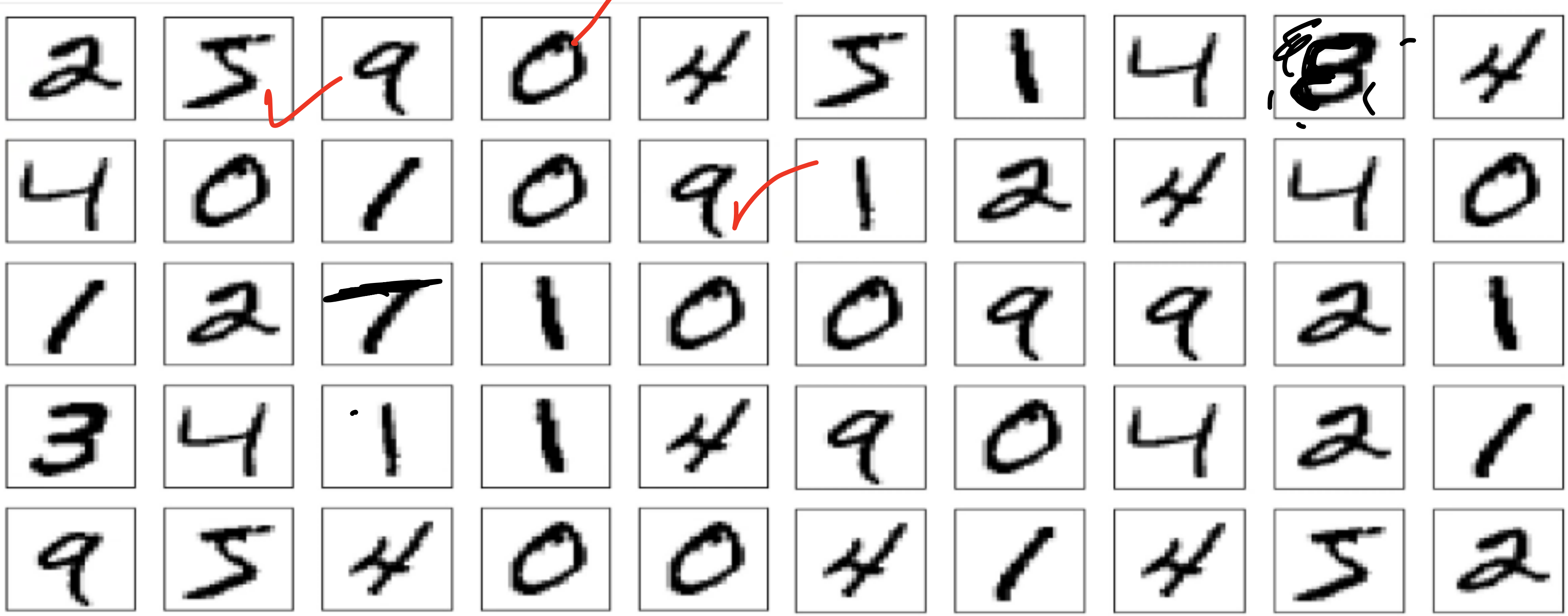
val_data = (x_val, y_val)

7

7

Multi-class → 10

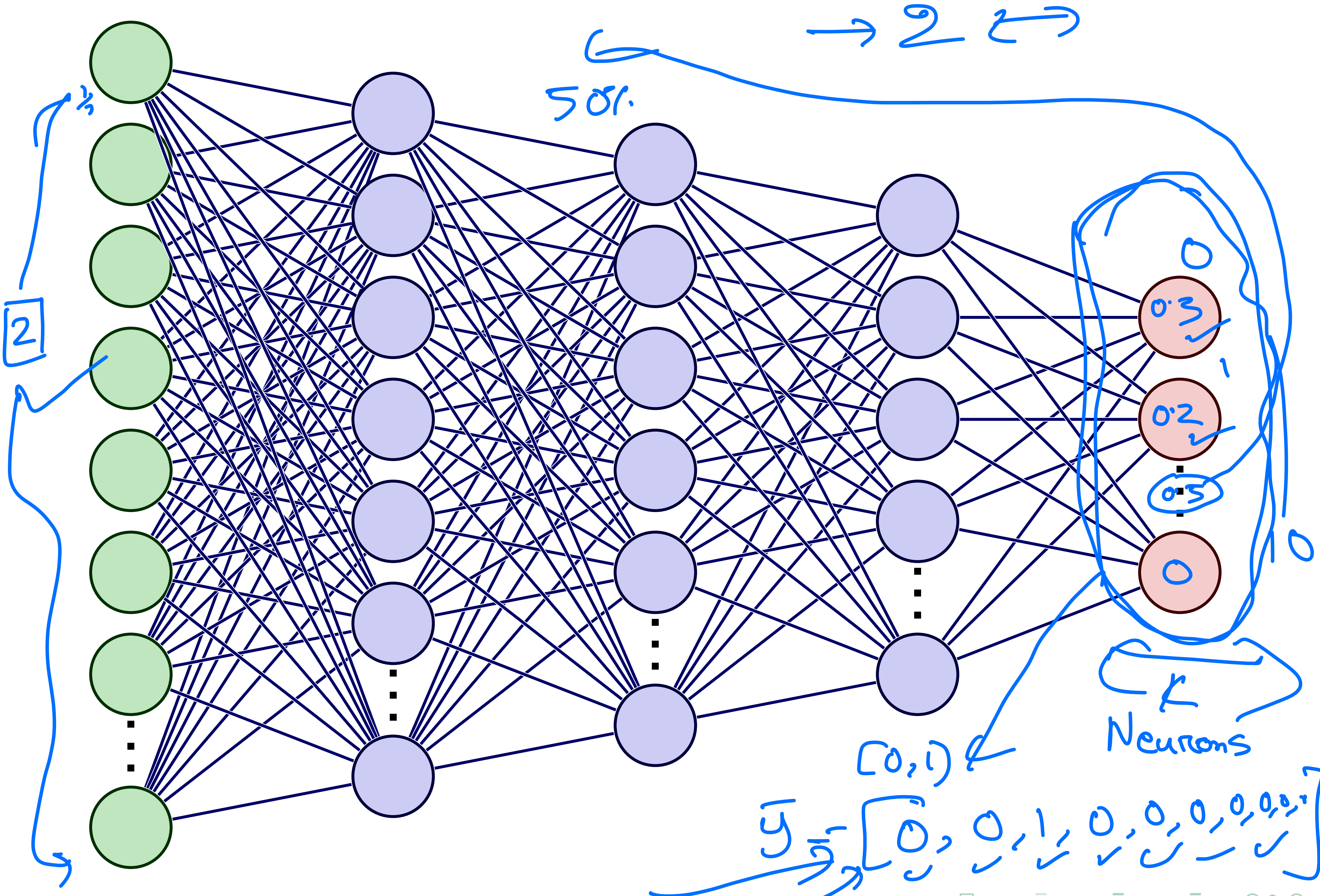
K → # of classes K=10



⟵ component ⟶

$$\bar{y} = \left[\begin{array}{cccccccccc} 0, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 0 \\ \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow \\ & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{array}\right]$$

→ OHE vector 1st component → tracking zero


→ 2 ⟵
50%
1/7
2
0
0.3
1
0.2
0.5
0
10
K
Neurons
[0,1)
y = [0, 0, 1, 0, 0, 0, 0, 0, 0, 0]

Implementation in tensorflow

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.utils import to_categorical
import matplotlib.pyplot as plt

(x_train, y_train), (x_test, y_test) = mnist.load_data()

x_train = x_train.reshape(60000, 784)
x_test = x_test.reshape(10000, 784)

x_train = x_train.astype('float32') / 255
x_test = x_test.astype('float32') / 255

# Convert the target data into one-hot encoding
y_train = to_categorical(y_train, 10)
y_test = to_categorical(y_test, 10)
```

one

Implementation in tensorflow (contd.)

Softmax, $\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}}$

Soft max scores

```
model = Sequential()
model.add(Dense(512, activation='relu', input_shape=(784,)))
model.add(Dense(256, activation='relu'))
model.add(Dense(128, activation='relu'))
model.add(Dense(10, activation='softmax'))

model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

model.summary()
```

sigmoid

→ [0, 1]

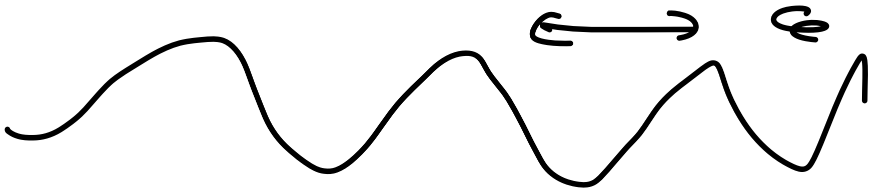
→ [0, 1]

⋮

⇒ [0, 1]

≠ 1

Step 2: Define model. You may need to perform cross-validation to find the optimal architecture.



Implementation in tensorflow (contd.)

```
history = model.fit(x_train, y_train, epochs=10,
batch_size=64, validation_split=0.3)

plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()
```

Step 3: Train model and analyze further. You can add more steps of postprocessing as needed.

HW4 hints

- P1. Identify the type of classification problem. Then use these ideas to code and answer the questions.
- P2. It is a regression problem. The question allows you to use any machine learning model to solve it. So, use whichever one you prefer. Try to identify the features and target and the rest should be pretty easy.

The CIFAR-10 image dataset

P1,

```python
from keras.datasets import cifar10
from keras.utils.np_utils import to_categorical
import matplotlib.pyplot as plt

# First time you run this it will download the data
(X_train, y_train), (X_test, y_test) = cifar10.load_data()

cifar_classes = ['airplane', 'automobile', 'bird', 'cat',
                 'deer', 'dog', 'frog', 'horse', 'ship', 'truck']
print('Example training images and their labels: ' + str([x[0] for x in y_train[0:5]]))
print('Corresponding classes for the labels: ' + str([cifar_classes[x[0]] for x in y_train[0:5]]))

f, axarr = plt.subplots(1, 5)
f.set_size_inches(16, 6)

for i in range(5):
    img = X_train[i]
    axarr[i].imshow(img)
plt.show()
```

Dataset description https://www.cs.toronto.edu/~kriz/cifar.html

Project proposal

What is my data?

March 2

- Submit your initial project proposal.
- Revise it based on our feedback
- Follow the guidelines posted in LMS
- Work on your project and keep us informed.
- Better to work on something that you find interesting.

1.

2

3