CAP 6619 Deep Learning
2024 Summer

Homework 2 [15 Pts, Due: June 2, 2024. Late Penalty: -2/day]

[If two homework submissions are found to be similar to each other, both submissions will receive 0 grade]

[Homework solutions must be submitted through Canvas. No email submission is accepted. If you have multiple files, please include all files as one zip file, and submit zip file online (only zip, pdf, or word files are allowed). You can always update your submissions. Only the latest version will be graded.]

Question 1 [2 pts]: Figure 1 shows some samples in a two-dimensional feature space (X_1, X_2). Each symbol (circle, triangle, plus, diamond) denotes a sample, and each shape denote one type of samples. Please design a neural network architecture which can learn to classify samples in Figure 3 into correct types.

- Show your network input dimension, number of hidden nodes (layers), and output node(s) [1 pt].
- Explain why you use this network architecture [1 pt]

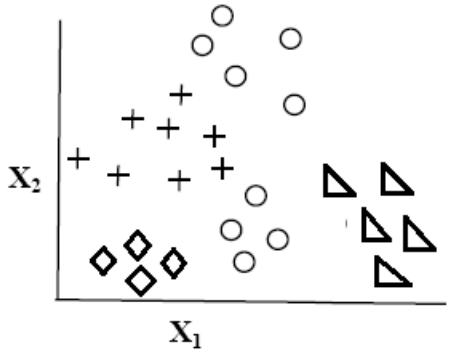
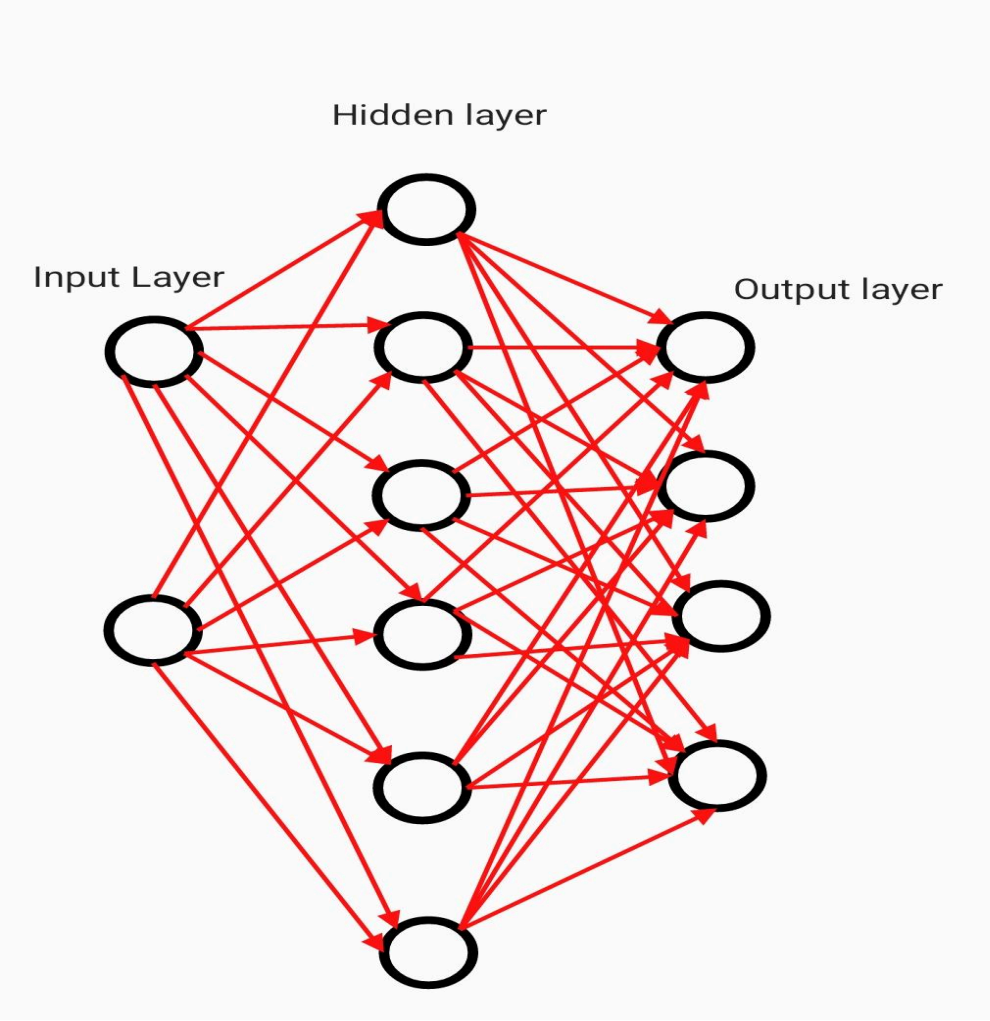


Figure 1

- Show your network input dimension, number of hidden nodes (layers), and output node(s) [1 pt].



- Explain why you use this network architecture [1 pt]

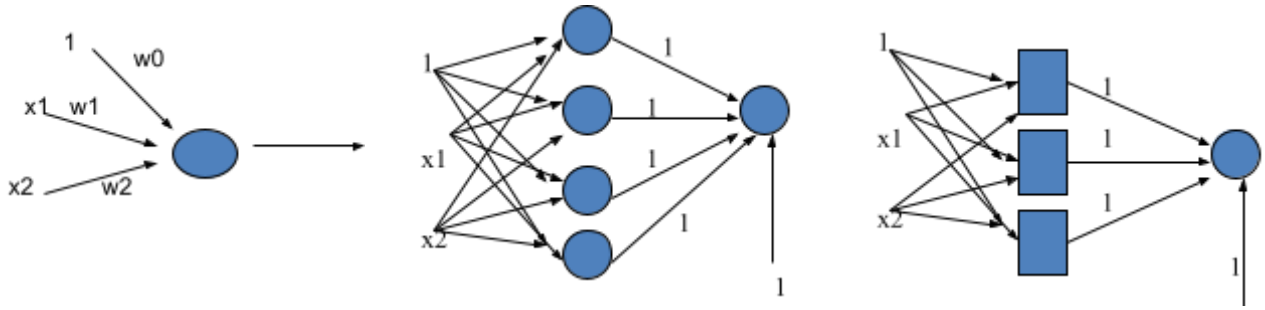
Input Layer
The data has two features so I used 2 neurons for the input layer.

Hidden Layer
The identifying characteristics of the shapes are the edges so I used only one hidden layer to identify the edges to predict the shape. The hidden layer has 6 neurons to identify the 4 shapes.

Output Layer
The total number of shapes are four, so the output consists of 4 neurons.

Question 2 [1 pt] Figure 2 shows three neural network structures. (a) is a single layer neural network, (b) is a one hidden layer neural network, and (c) is a two hidden layer neural network, where each rectangle box denotes a one hidden layer network similar to (b).

- Please draw decision regions for each network, respectively [0.5 pt],
- Explain how the networks learn to classify samples into different categories [0.5 pt]

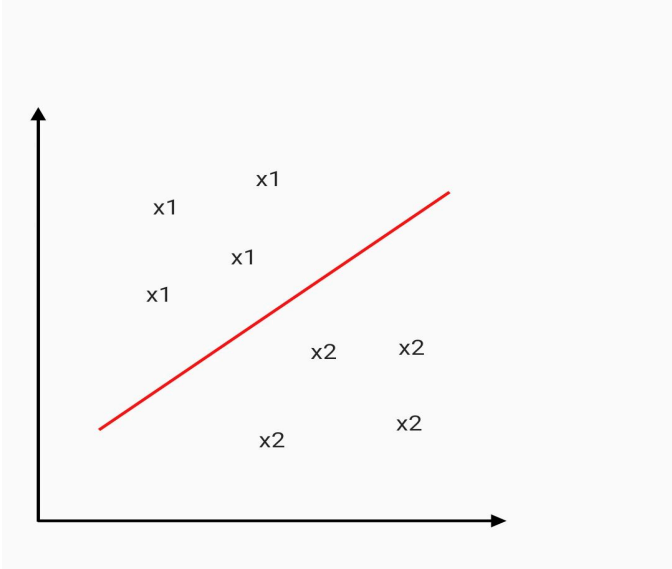


(a) (b) (c)

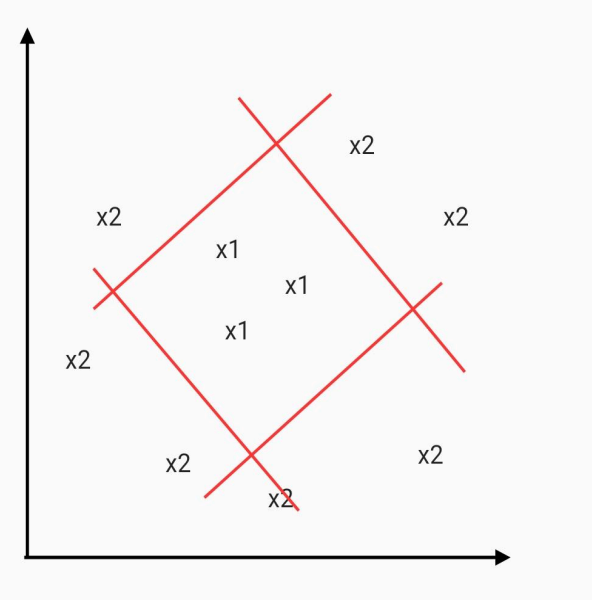
Figure 2: Feedforward neural networks

- Please draw decision regions for each network, respectively [0.5 pt]

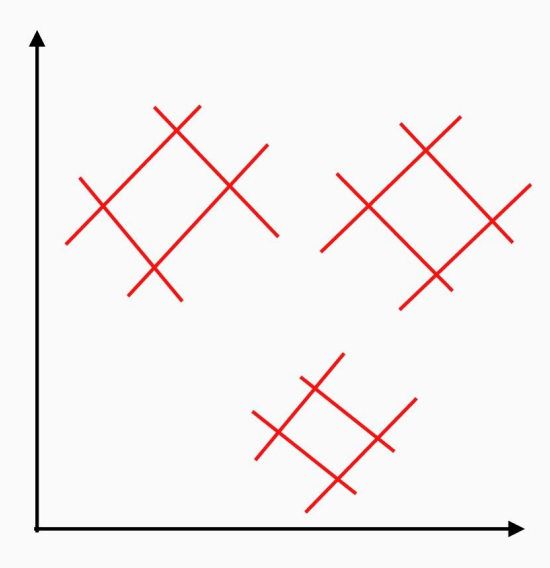
a.



b.



c.



- Explain how the networks learn to classify samples into different categories [0.5 pt]

For a,
It is a single layer neural network. The decision boundary is a hyperplane separating the two.

For b,

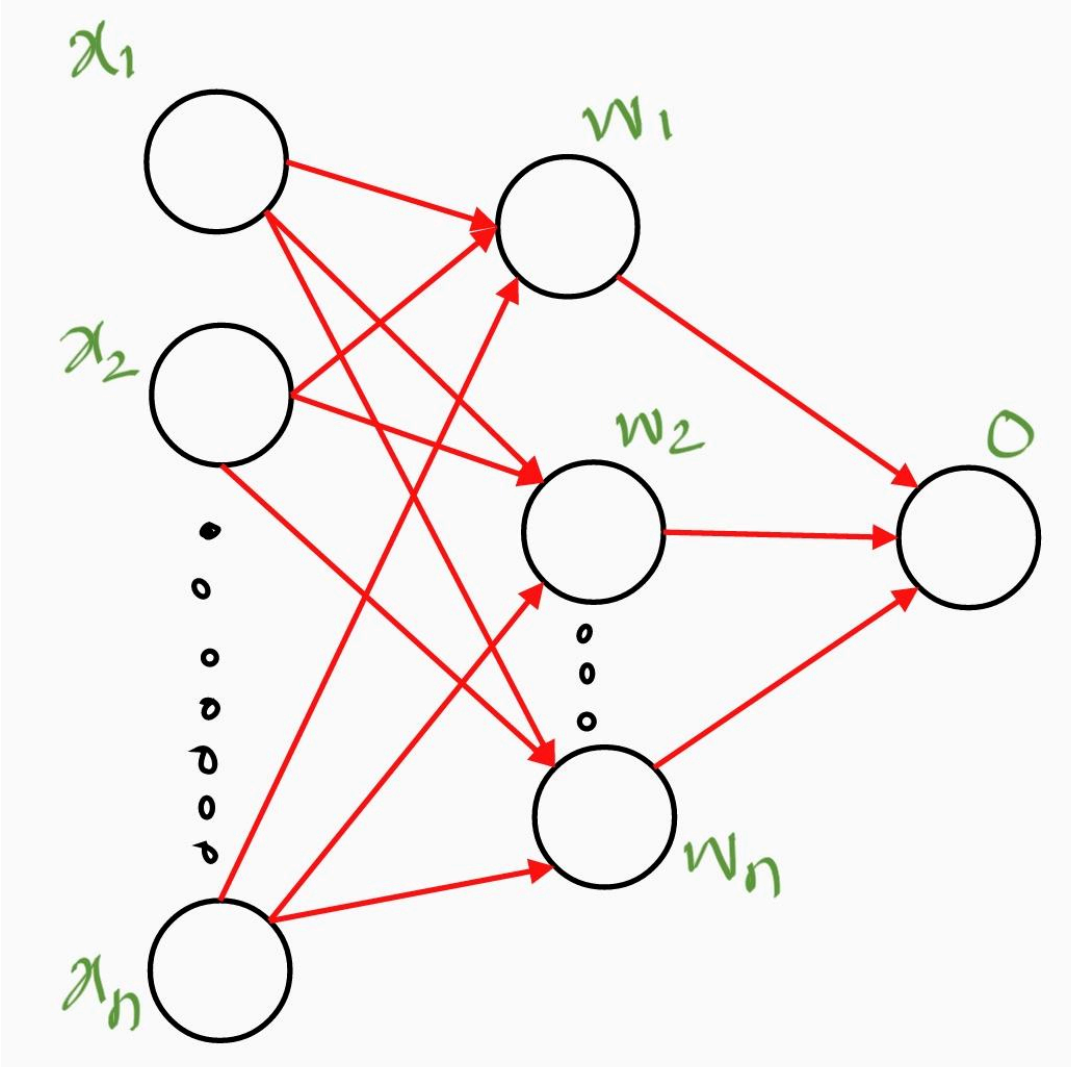
It consists of a hidden layer, so the decision boundary is convex. Since, there's 4 neurons within the hidden layer, each neuron will give a line bounding the decision surface.

For c,
The 3 hidden layers converge into 1 in the final layer and each hidden layer predicts an individual convex region.

Question 3 [2 pts]. Given a neuron with *n* dimensional input features [x_1, x_2, …, x_n], assuming the output from the neuron is defined by

$$o = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3 + \ldots + w_n x_n$$

- Please draw the structure of the neuron (show input, output, and the weight) [0.5 pt]
- Given a batch of N training instances (x(1), d(1)), (x(2), d(2)), …, (x(N), d(N)), where d(i) denotes label of instance x(i), show gradient descent learning objective function of the neuron [0.5 pt]
- Derive gradient descent learning weight updating rule for weight w_i [1 pt].



Given,

$$O = \omega_0 + w_1 x_1 + \cdots w_n x_n$$

Here, for an example (x, d), the error $e(w)$ of the network is,

$$e(w) = d - O(x) = d - \sum_{j=0}^{m} x_j w_j$$

$$E(w) = \frac{1}{2} e^2$$

$$-(\text{gradient of } E(w)) = -\left[\frac{\partial E}{\partial w_1}, \cdots, \frac{\partial E}{\partial w_n}\right]$$

$$w(k+1) = w(k) - \eta\,(\text{gradient of } E(w))$$

Gradient :

$$\nabla E(w) = [\partial E/\partial \omega_0 \cdots \partial E/\partial \omega_n]$$

$$\Delta w = -\eta\, \nabla E(w)$$

$$= -\eta\, \partial E/\partial \omega_i$$

$$= \eta \frac{\partial}{\partial w_i} \frac{1}{2} \sum_n (d(n) - o(n))^2$$

$$= \eta \frac{\partial}{\partial w_i} \frac{1}{2} \sum_n (d(n) - \sum w_i x_i(n))^2$$

$$= \eta \sum_n (d(n) - o(n)) x_i(n)$$

$$w_i(k+1) = w_i(k) + \Delta w_i$$

$$= w_i(k) + \eta \sum_n d(n) - o(n)) x_i(n)$$

Question 4 [2 pts]: In Figure 3, a one hidden layer neural network is trained to predict an input as “dog” or “cat”. For each input, if the label of the input is “dog”, its output from node “e” is expected to be 1, otherwise the output is expected to be 0. Similarly, if the label of the input is “Cat”, its output from node “f” is expected to be 1, otherwise the output from node “f” is 0 (using sigmoid activation function, and parameter a=1).
Given an input [x0, x1, x2]=[1, 0.5, 0.5] which is labeled as “dog”, where x0 denotes bias,

- Calculate its actual output from node “e” and “f”, respectively. [0.5 pt]
- Calculate network error of the instance. [0.5 pt]
- Calculate local gradient of the instance with respect to node “e” and “f”, respectively. [0.5 pt]
- Calculate local gradient of the instance with respect to node “a” and “b”, respectively [0.5 pt]

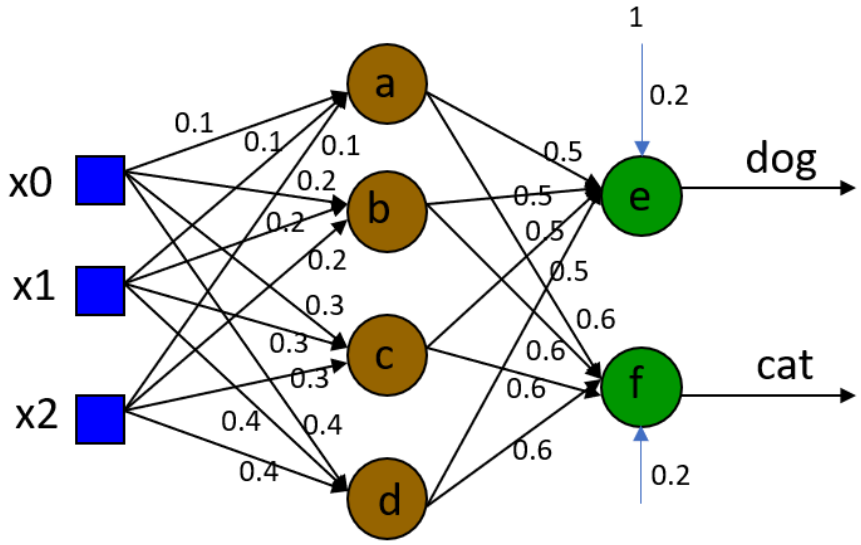


Figure 3: A one hidden layer neural network

- Calculate its actual output from node “e” and “f”, respectively. [0.5 pt]

The sigmoid function can be defined as,
$\sigma(x) = 1 / (1 - e^{-x})$
For this problem,
$y = 1 / (1 - e^{-wx + b})$, where,

w = weight
x = input
b = bias

Given,

x0 = 1
x1 = 0.5
x2= 0.5
For a,
$y(a) = 1 / (1 - e^{-(1 + 0.5 * w + 0.5 * w)}) = 0.75$

Similarly,
$y(b) = 1 / (1 - e^{-(1 + 0.5 * w + 0.5 * w)}) = 0.77$
$y(c) = 1 / (1 - e^{-(1 + 0.5 * w + 0.5 * w)}) = 0.79$
$y(d) = 1 / (1 - e^{-(1 + 0.5 * w + 0.5 * w)}) = 0.80$

For,
$y(e) = 1 / (1 - e^{-(1 + a * w + b * w + c * w + d * w)}) = 0.85$
$y(f) = 1 / (1 - e^{-(1 + a * w + b * w + c * w + d * w)}) = 0.89$

Since, y(f) > y(e), the output is a cat.

- Calculate network error of the instance. [0.5 pt]

The error can be defined as

$E(n) = \sum (\text{true value} - \text{predicted value})^2 / 2$

The error of this network is,
$((1 - 0.85)^2 + (0 - 0.89)^2)/2 = 0.41$

- Calculate local gradient of the instance with respect to node “e” and “f”, respectively. [0.5 pt]

For an output layer,
$\delta_k = \sigma_k (1 - \sigma_k) (d_k - o_k)$

So,
$\delta(e) = 0.185$
$\delta(f) = - 0.0886$

- Calculate local gradient of the instance with respect to node “a” and “b”, respectively

For hidden layer,
$\delta_h = \sigma_h (1 - \sigma_h) \sum w_{hk} \delta_k$

$\delta(a) = 0.0082$
$\delta(b) = - 0.0078$

Question 5 [3 pts]: Given the following feedforward neural network with one hidden layer and one output layer, assuming the network initial weights are

$$\begin{bmatrix} w_{0a} \\ w_{1a} \\ w_{2a} \end{bmatrix} = \begin{bmatrix} 1.0 \\ 1.0 \\ 1.0 \end{bmatrix}; \quad \begin{bmatrix} w_{0b} \\ w_{1b} \\ w_{2b} \end{bmatrix} = \begin{bmatrix} 1.0 \\ 1.0 \\ 1.0 \end{bmatrix}; \quad \begin{bmatrix} w_{0c} \\ w_{ac} \\ w_{bc} \end{bmatrix} = \begin{bmatrix} 1.0 \\ 1.0 \\ 1.0 \end{bmatrix}.$$

All nodes use a sigmoid activation function with value a=1.0, and the learning rate η is set to 0.1. The expected output is *1* if an instance's class label is "True", otherwise, the expected output is *0*. Given the following three instances I_1, I_2, I_3 which are labeled as "True", "True", and "False", respectively,

- Please calculate actual outputs of each instance from the network? [1 **pt**]
- Calculate mean squared error of the network with respect to the three instances? [**1 pt**]
- Assuming instance I_1 is fed to the network to update the weight, please update the network weight (using backpropogation rule) by using this instance (list major steps and results). [**1 pt**]

$$I_1 = \begin{bmatrix} 1.0 \\ 1.0 \\ 0.5 \end{bmatrix}; \quad I_2 = \begin{bmatrix} 1.0 \\ 0.0 \\ 1.0 \end{bmatrix}; \quad I_3 = \begin{bmatrix} 1.0 \\ 0.5 \\ 0.5 \end{bmatrix}$$

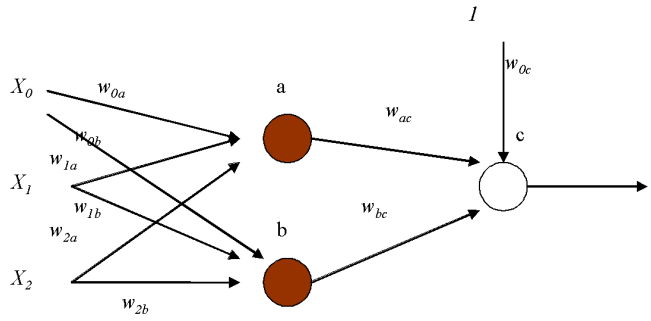


Figure 4: A hidden layer neural network

Given,

$$\begin{bmatrix} w_{0a} \\ w_{1a} \\ w_{2a} \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \quad \begin{bmatrix} w_{0b} \\ w_{1b} \\ w_{2b} \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \quad \begin{bmatrix} w_{0c} \\ w_{1c} \\ w_{2c} \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

The sigmoid function,

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

for neural networks with sigmoid activation function,

$$y = \frac{1}{1+e^{-wx+b}}, \text{ where,}$$

w = weight
b = bias.

For a,

$$a = \frac{1}{1+e^{-(x_0 w_{0a} + x_1 w_{1a} + x_2 w_{2a})}}$$

For b,

$$b = \frac{1}{1 + e^{-(\gamma_0 w_{0b} + \gamma_1 w_{1b} + \gamma_2 w_{2b})}}$$

For c,

$$c = \frac{1}{1 + e^{-(w_{0c} + a w_{ac} + b w_{bc})}}$$

For $t_1 = \begin{bmatrix} 1 \\ 1 \\ 0.5 \end{bmatrix}$, $a = 0.924$, $b = 0.924$, $c = 0.945$

For $t_2 = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$, $a = 0.881$, $b = 0.881$, $c = 0.941$

For $t_3 = \begin{bmatrix} 1 \\ 0.5 \\ 0.5 \end{bmatrix}$, $a = 0.881$, $b = 0.881$, $c = 0.941$

Network error $= \frac{1}{2} \sum_{i} (p_{true} - p_{predicted})^2$

For instance l_1, true label = 1

$e = \frac{1}{2}(1 - 0.945)^2 = 0.0015$

For instance l_2, true label = 1

$e = \frac{1}{2}(1 - 0.941)^2 = 0.00177$

For instance l_3, true label = 0

$e = \frac{1}{2}(0 - 0.941)^2 = 0.4423$

For instance $I_1 = \begin{bmatrix} 1 \\ 1 \\ 0.5 \end{bmatrix}$, $\eta = 0.1$

For output layer,

$$\delta_K = o_K (1 - o_K)(d_K - o_K)$$

For hidden layer,

$$\delta_h = o_h (1 - o_h) \sum w_{hK} \delta_K$$

$$w_{ij} = w_{ij} + \Delta w_{ij}, \quad \Delta w_{ij} = \eta \delta_j x_{ij}$$

Now, $\delta_c = 0.0028$, ~~$\Delta w_{cj} =$~~

$\delta_b = 0.000199$, ~~$\Delta w_b =$~~

$\delta_a = 0.000199$, ~~$\Delta w_a =$~~

$\Delta w_{0c} = 0.00028$, $w_{0c} = 1.00028$

$\Delta w_{ac} = 0.00026$, $w_{ac} = 1.00026$

$\Delta w_{bc} = 0.00028$, $w_{bc} = 1.00028$

$\Delta w_{0a} = 1.99 \times 10^{-5}$, $w_{0a} = 1.000019$

$\Delta w_{1a} = 1.99 \times 10^{-5}$, $w_{1a} = 1.00001$

$\Delta w_{2a} = 9.94 \times 10^{-6}$, $w_{2a} = 1.0000099$

$\Delta w_{0b} = 1.99 \times 10^{-5}$, $w_{0b} = 1.000019$

$\Delta w_{1b} = 1.99 \times 10^{-5}$, $w_{1b} = 1.00001$

$\Delta w_{2b} = 9.94 \times 10^{-6}$, $w_{2b} = 1.0000099$

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

import tensorflow as tf
import keras
from keras.models import Sequential
from keras.layers import Activation, Flatten, Dense, Dropout
from tensorflow.keras import layers
from tensorflow.keras.utils import load_img
from tensorflow.keras.utils import img_to_array
from tensorflow.keras.layers import BatchNormalization


from sklearn.model_selection import StratifiedKFold


from PIL import Image as im
from sklearn.model_selection import train_test_split
```

```
2024-06-02 21:55:46.967405: I tensorflow/core/platform/cpu_feature_guard.cc:
193] This TensorFlow binary is optimized with oneAPI Deep Neural Network Lib
rary (oneDNN) to use the following CPU instructions in performance-critical
operations:  AVX2 AVX_VNNI FMA
To enable them in other operations, rebuild TensorFlow with the appropriate
compiler flags.
2024-06-02 21:55:47.040811: I tensorflow/core/util/port.cc:104] oneDNN custo
m operations are on. You may see slightly different numerical results due to
floating-point round-off errors from different computation orders. To turn t
hem off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
2024-06-02 21:55:47.391770: W tensorflow/compiler/xla/stream_executor/platfo
rm/default/dso_loader.cc:64] Could not load dynamic library 'libnvinfer.so.7
'; dlerror: libnvinfer.so.7: cannot open shared object file: No such file or
directory; LD_LIBRARY_PATH: /usr/local/cuda-12.3/lib64:/usr/local/cuda-12.3/
lib64:
2024-06-02 21:55:47.391812: W tensorflow/compiler/xla/stream_executor/platfo
rm/default/dso_loader.cc:64] Could not load dynamic library 'libnvinfer_plug
in.so.7'; dlerror: libnvinfer_plugin.so.7: cannot open shared object file: N
o such file or directory; LD_LIBRARY_PATH: /usr/local/cuda-12.3/lib64:/usr/l
ocal/cuda-12.3/lib64:
2024-06-02 21:55:47.391815: W tensorflow/compiler/tf2tensorrt/utils/py_util
s.cc:38] TF-TRT Warning: Cannot dlopen some TensorRT libraries. If you would
like to use Nvidia GPU with TensorRT, please make sure the missing libraries
mentioned above are installed properly.
```

In [2]:
```
from os import listdir
from numpy import asarray
from numpy import save
from matplotlib.image import imread
```

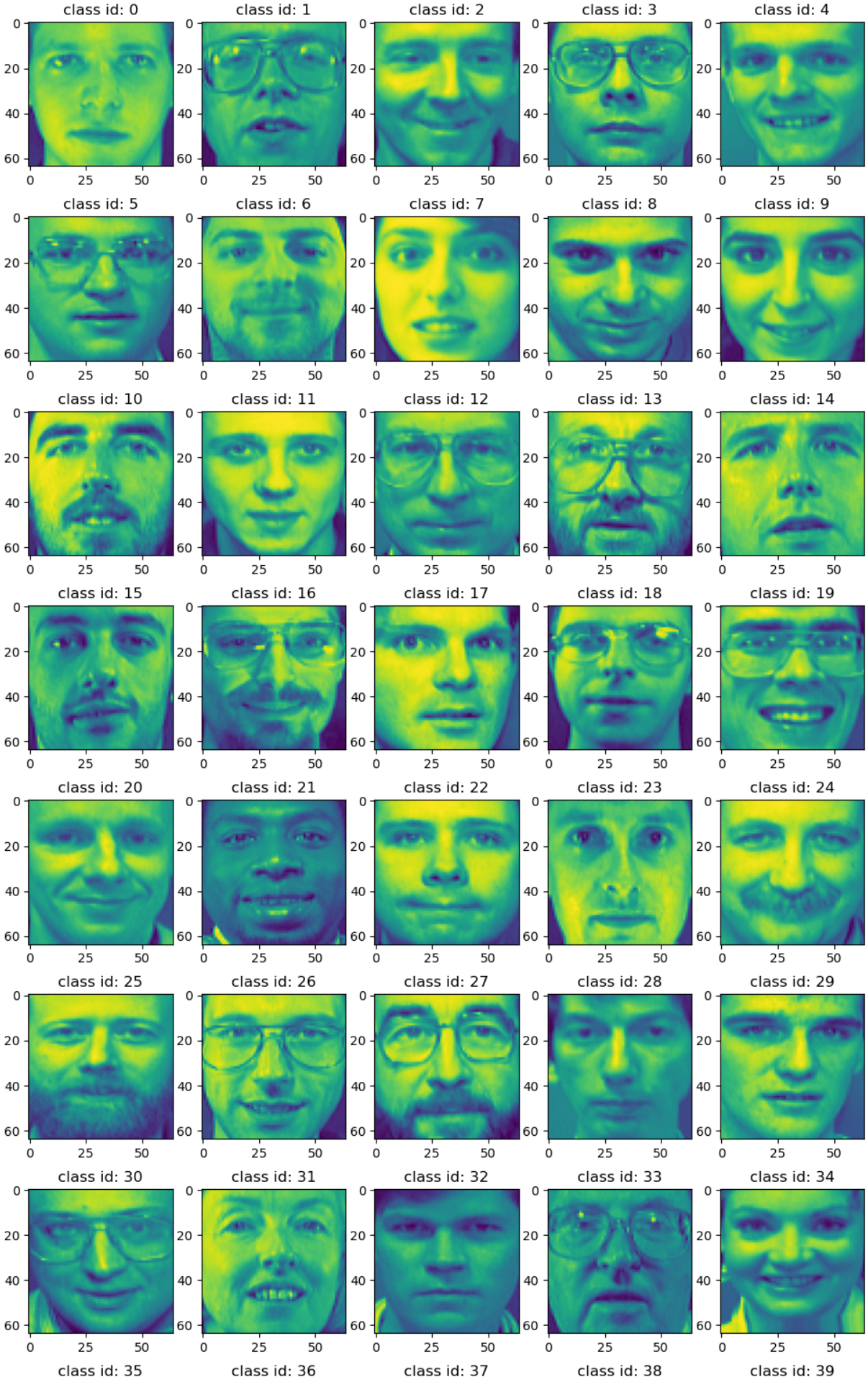
Question 6 [2 pts] The Dense Layer Classification [Notebook, html] in the Canvas shows detailed procedures about how to

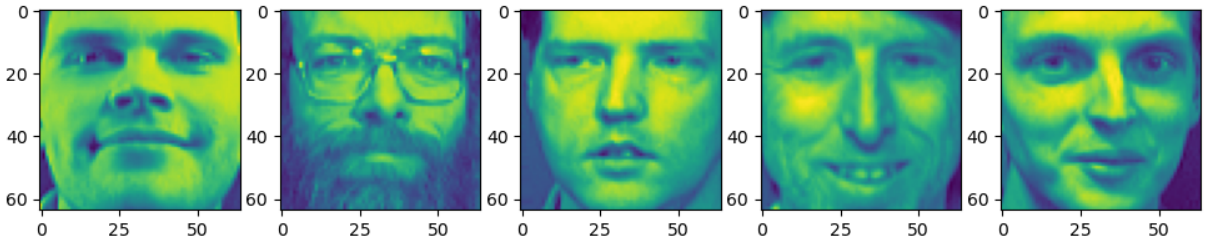
use Oivetti face dataset (from AT&T) to train Neural Network classifiers for face classification. The dataset in the Canvas also provides “olivetti_faces.npy” and “olivetti_faces_target.npy”, which includes 400 faces in 40 classes (40 different person). Please implement following face recognition task using Neural Networks.

Please show at least one face images for each class in the Oivetti face dataset100. Randomly split the dataset into 60% training and 40% test samples. Train a one-hidden layer neural network with 10 hidden nodes. Report the classification accuracy of the classifier on the test set [1 pt]

In [3]:
```
x = np.load("olivetti_faces.npy")
y = np.load("olivetti_faces_target.npy")
```

In [4]:
```
plt.figure(figsize = [12,22])
i = 0

for j in range(0,40):

    arr = x[i]

    plt.subplot(8,5,j+1)
    plt.imshow(arr)
    plt.title( "class id: {}".format(y[i]))

    i = i+10
```


class id: 0
class id: 1
class id: 2
class id: 3
class id: 4
class id: 5
class id: 6
class id: 7
class id: 8
class id: 9
class id: 10
class id: 11
class id: 12
class id: 13
class id: 14
class id: 15
class id: 16
class id: 17
class id: 18
class id: 19
class id: 20
class id: 21
class id: 22
class id: 23
class id: 24
class id: 25
class id: 26
class id: 27
class id: 28
class id: 29
class id: 30
class id: 31
class id: 32
class id: 33
class id: 34
class id: 35
class id: 36
class id: 37
class id: 38
class id: 39



In [5]:
```
x_train, x_test, y_train, y_test = train_test_split(x,y , random_state=10,te
```

In [6]:
```
model = keras.Sequential([
    keras.layers.Flatten(input_shape=(64, 64, 1),name="Input"),
    keras.layers.Dense(10, activation='sigmoid',name="Hidden"),
    keras.layers.Dense(40, name="Output"),
])
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 Input (Flatten)             (None, 4096)              0

 Hidden (Dense)              (None, 10)                40970

 Output (Dense)              (None, 40)                440

=================================================================
Total params: 41,410
Trainable params: 41,410
Non-trainable params: 0
_________________________________________________________________
2024-06-02 21:55:50.198841: E tensorflow/compiler/xla/stream_executor/cuda/c
uda_driver.cc:267] failed call to cuInit: CUDA_ERROR_UNKNOWN: unknown error
2024-06-02 21:55:50.198857: I tensorflow/compiler/xla/stream_executor/cuda/c
uda_diagnostics.cc:169] retrieving CUDA diagnostic information for host: pop
-os
2024-06-02 21:55:50.198860: I tensorflow/compiler/xla/stream_executor/cuda/c
uda_diagnostics.cc:176] hostname: pop-os
2024-06-02 21:55:50.198954: I tensorflow/compiler/xla/stream_executor/cuda/c
uda_diagnostics.cc:200] libcuda reported version is: 550.54.14
2024-06-02 21:55:50.198966: I tensorflow/compiler/xla/stream_executor/cuda/c
uda_diagnostics.cc:204] kernel reported version is: 550.54.14
2024-06-02 21:55:50.198968: I tensorflow/compiler/xla/stream_executor/cuda/c
uda_diagnostics.cc:310] kernel version seems to match DSO: 550.54.14
2024-06-02 21:55:50.199146: I tensorflow/core/platform/cpu_feature_guard.cc:
193] This TensorFlow binary is optimized with oneAPI Deep Neural Network Lib
rary (oneDNN) to use the following CPU instructions in performance-critical
operations:  AVX2 AVX_VNNI FMA
To enable them in other operations, rebuild TensorFlow with the appropriate
compiler flags.
```

In [7]:
```
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits
              metrics=['accuracy'])
```

In [8]:
```
history=model.fit(x_train, y_train, validation_split=0.2, epochs=300,verbose
```

```
Epoch 1/300
6/6 [==============================] - 0s 19ms/step - loss: 3.7239 - accurac
y: 0.0104 - val_loss: 3.6896 - val_accuracy: 0.0208
Epoch 2/300
6/6 [==============================] - 0s 4ms/step - loss: 3.6883 - accurac
y: 0.0469 - val_loss: 3.6907 - val_accuracy: 0.0000e+00
Epoch 3/300
6/6 [==============================] - 0s 4ms/step - loss: 3.6880 - accurac
y: 0.0365 - val_loss: 3.6913 - val_accuracy: 0.0000e+00
Epoch 4/300
6/6 [==============================] - 0s 5ms/step - loss: 3.6876 - accurac
y: 0.0365 - val_loss: 3.6921 - val_accuracy: 0.0000e+00
Epoch 5/300
6/6 [==============================] - 0s 4ms/step - loss: 3.6871 - accurac
y: 0.0365 - val_loss: 3.6944 - val_accuracy: 0.0208
Epoch 6/300
6/6 [==============================] - 0s 4ms/step - loss: 3.6833 - accurac
y: 0.0417 - val_loss: 3.7066 - val_accuracy: 0.0208
Epoch 7/300
6/6 [==============================] - 0s 4ms/step - loss: 3.6786 - accurac
y: 0.0312 - val_loss: 3.7102 - val_accuracy: 0.0208
Epoch 8/300
6/6 [==============================] - 0s 4ms/step - loss: 3.6835 - accurac
y: 0.0573 - val_loss: 3.6993 - val_accuracy: 0.0208
Epoch 9/300
6/6 [==============================] - 0s 4ms/step - loss: 3.6741 - accurac
y: 0.0469 - val_loss: 3.7283 - val_accuracy: 0.0208
Epoch 10/300
6/6 [==============================] - 0s 4ms/step - loss: 3.6732 - accurac
y: 0.0312 - val_loss: 3.7062 - val_accuracy: 0.0208
Epoch 11/300
6/6 [==============================] - 0s 4ms/step - loss: 3.6719 - accurac
y: 0.0365 - val_loss: 3.7237 - val_accuracy: 0.0208
Epoch 12/300
6/6 [==============================] - 0s 5ms/step - loss: 3.6643 - accurac
y: 0.0312 - val_loss: 3.7077 - val_accuracy: 0.0208
Epoch 13/300
6/6 [==============================] - 0s 4ms/step - loss: 3.6657 - accurac
y: 0.0417 - val_loss: 3.7151 - val_accuracy: 0.0208
Epoch 14/300
6/6 [==============================] - 0s 4ms/step - loss: 3.6610 - accurac
y: 0.0312 - val_loss: 3.7136 - val_accuracy: 0.0208
Epoch 15/300
6/6 [==============================] - 0s 4ms/step - loss: 3.6556 - accurac
y: 0.0365 - val_loss: 3.7141 - val_accuracy: 0.0208
Epoch 16/300
6/6 [==============================] - 0s 4ms/step - loss: 3.6512 - accurac
y: 0.0521 - val_loss: 3.7236 - val_accuracy: 0.0208
Epoch 17/300
6/6 [==============================] - 0s 4ms/step - loss: 3.6459 - accurac
y: 0.0365 - val_loss: 3.7115 - val_accuracy: 0.0208
Epoch 18/300
6/6 [==============================] - 0s 4ms/step - loss: 3.6371 - accurac
y: 0.0365 - val_loss: 3.7139 - val_accuracy: 0.0208
Epoch 19/300
6/6 [==============================] - 0s 4ms/step - loss: 3.6310 - accurac
```

```
y: 0.0469 - val_loss: 3.7098 - val_accuracy: 0.0208
Epoch 20/300
6/6 [==============================] - 0s 4ms/step - loss: 3.6244 - accurac
y: 0.0469 - val_loss: 3.7117 - val_accuracy: 0.0208
Epoch 21/300
6/6 [==============================] - 0s 4ms/step - loss: 3.6195 - accurac
y: 0.0469 - val_loss: 3.7122 - val_accuracy: 0.0208
Epoch 22/300
6/6 [==============================] - 0s 4ms/step - loss: 3.6160 - accurac
y: 0.0417 - val_loss: 3.7048 - val_accuracy: 0.0208
Epoch 23/300
6/6 [==============================] - 0s 4ms/step - loss: 3.6092 - accurac
y: 0.0677 - val_loss: 3.7110 - val_accuracy: 0.0208
Epoch 24/300
6/6 [==============================] - 0s 4ms/step - loss: 3.6053 - accurac
y: 0.0469 - val_loss: 3.7098 - val_accuracy: 0.0208
Epoch 25/300
6/6 [==============================] - 0s 4ms/step - loss: 3.6025 - accurac
y: 0.0521 - val_loss: 3.7053 - val_accuracy: 0.0208
Epoch 26/300
6/6 [==============================] - 0s 4ms/step - loss: 3.5958 - accurac
y: 0.0573 - val_loss: 3.7134 - val_accuracy: 0.0208
Epoch 27/300
6/6 [==============================] - 0s 4ms/step - loss: 3.5972 - accurac
y: 0.0469 - val_loss: 3.7016 - val_accuracy: 0.0208
Epoch 28/300
6/6 [==============================] - 0s 4ms/step - loss: 3.5850 - accurac
y: 0.0573 - val_loss: 3.7131 - val_accuracy: 0.0000e+00
Epoch 29/300
6/6 [==============================] - 0s 4ms/step - loss: 3.5835 - accurac
y: 0.0729 - val_loss: 3.6979 - val_accuracy: 0.0208
Epoch 30/300
6/6 [==============================] - 0s 4ms/step - loss: 3.5849 - accurac
y: 0.0677 - val_loss: 3.7243 - val_accuracy: 0.0208
Epoch 31/300
6/6 [==============================] - 0s 4ms/step - loss: 3.5796 - accurac
y: 0.0833 - val_loss: 3.6958 - val_accuracy: 0.0208
Epoch 32/300
6/6 [==============================] - 0s 4ms/step - loss: 3.5777 - accurac
y: 0.0729 - val_loss: 3.7135 - val_accuracy: 0.0208
Epoch 33/300
6/6 [==============================] - 0s 4ms/step - loss: 3.5710 - accurac
y: 0.0573 - val_loss: 3.7032 - val_accuracy: 0.0208
Epoch 34/300
6/6 [==============================] - 0s 4ms/step - loss: 3.5691 - accurac
y: 0.0573 - val_loss: 3.7088 - val_accuracy: 0.0208
Epoch 35/300
6/6 [==============================] - 0s 4ms/step - loss: 3.5640 - accurac
y: 0.0573 - val_loss: 3.7034 - val_accuracy: 0.0208
Epoch 36/300
6/6 [==============================] - 0s 4ms/step - loss: 3.5601 - accurac
y: 0.0625 - val_loss: 3.6949 - val_accuracy: 0.0208
Epoch 37/300
6/6 [==============================] - 0s 4ms/step - loss: 3.5571 - accurac
y: 0.0781 - val_loss: 3.7079 - val_accuracy: 0.0208
Epoch 38/300
```

```
6/6 [==============================] - 0s 4ms/step - loss: 3.5533 - accurac
y: 0.0677 - val_loss: 3.6943 - val_accuracy: 0.0208
Epoch 39/300
6/6 [==============================] - 0s 4ms/step - loss: 3.5505 - accurac
y: 0.0781 - val_loss: 3.7103 - val_accuracy: 0.0208
Epoch 40/300
6/6 [==============================] - 0s 4ms/step - loss: 3.5463 - accurac
y: 0.0833 - val_loss: 3.6922 - val_accuracy: 0.0208
Epoch 41/300
6/6 [==============================] - 0s 5ms/step - loss: 3.5440 - accurac
y: 0.0781 - val_loss: 3.6969 - val_accuracy: 0.0208
Epoch 42/300
6/6 [==============================] - 0s 4ms/step - loss: 3.5398 - accurac
y: 0.0729 - val_loss: 3.7003 - val_accuracy: 0.0208
Epoch 43/300
6/6 [==============================] - 0s 4ms/step - loss: 3.5351 - accurac
y: 0.0833 - val_loss: 3.6959 - val_accuracy: 0.0208
Epoch 44/300
6/6 [==============================] - 0s 4ms/step - loss: 3.5319 - accurac
y: 0.0781 - val_loss: 3.6987 - val_accuracy: 0.0208
Epoch 45/300
6/6 [==============================] - 0s 4ms/step - loss: 3.5335 - accurac
y: 0.0833 - val_loss: 3.6906 - val_accuracy: 0.0208
Epoch 46/300
6/6 [==============================] - 0s 4ms/step - loss: 3.5294 - accurac
y: 0.0677 - val_loss: 3.6974 - val_accuracy: 0.0208
Epoch 47/300
6/6 [==============================] - 0s 4ms/step - loss: 3.5260 - accurac
y: 0.0885 - val_loss: 3.6952 - val_accuracy: 0.0208
Epoch 48/300
6/6 [==============================] - 0s 4ms/step - loss: 3.5190 - accurac
y: 0.0729 - val_loss: 3.6901 - val_accuracy: 0.0208
Epoch 49/300
6/6 [==============================] - 0s 4ms/step - loss: 3.5167 - accurac
y: 0.0990 - val_loss: 3.6938 - val_accuracy: 0.0208
Epoch 50/300
6/6 [==============================] - 0s 4ms/step - loss: 3.5141 - accurac
y: 0.1094 - val_loss: 3.6924 - val_accuracy: 0.0208
Epoch 51/300
6/6 [==============================] - 0s 4ms/step - loss: 3.5112 - accurac
y: 0.0938 - val_loss: 3.6915 - val_accuracy: 0.0208
Epoch 52/300
6/6 [==============================] - 0s 4ms/step - loss: 3.5079 - accurac
y: 0.1146 - val_loss: 3.6960 - val_accuracy: 0.0208
Epoch 53/300
6/6 [==============================] - 0s 4ms/step - loss: 3.5051 - accurac
y: 0.0990 - val_loss: 3.6820 - val_accuracy: 0.0208
Epoch 54/300
6/6 [==============================] - 0s 4ms/step - loss: 3.5020 - accurac
y: 0.1354 - val_loss: 3.6993 - val_accuracy: 0.0208
Epoch 55/300
6/6 [==============================] - 0s 4ms/step - loss: 3.5009 - accurac
y: 0.1094 - val_loss: 3.6787 - val_accuracy: 0.0417
Epoch 56/300
6/6 [==============================] - 0s 4ms/step - loss: 3.5039 - accurac
y: 0.1146 - val_loss: 3.7037 - val_accuracy: 0.0208
```

```
Epoch 57/300
6/6 [==============================] - 0s 4ms/step - loss: 3.4994 - accurac
y: 0.0990 - val_loss: 3.6880 - val_accuracy: 0.0208
Epoch 58/300
6/6 [==============================] - 0s 4ms/step - loss: 3.4892 - accurac
y: 0.0990 - val_loss: 3.6926 - val_accuracy: 0.0208
Epoch 59/300
6/6 [==============================] - 0s 4ms/step - loss: 3.4875 - accurac
y: 0.0885 - val_loss: 3.6917 - val_accuracy: 0.0208
Epoch 60/300
6/6 [==============================] - 0s 4ms/step - loss: 3.4839 - accurac
y: 0.0938 - val_loss: 3.6907 - val_accuracy: 0.0208
Epoch 61/300
6/6 [==============================] - 0s 4ms/step - loss: 3.4871 - accurac
y: 0.1146 - val_loss: 3.6852 - val_accuracy: 0.0208
Epoch 62/300
6/6 [==============================] - 0s 4ms/step - loss: 3.4775 - accurac
y: 0.1042 - val_loss: 3.6772 - val_accuracy: 0.0208
Epoch 63/300
6/6 [==============================] - 0s 4ms/step - loss: 3.4740 - accurac
y: 0.1198 - val_loss: 3.6843 - val_accuracy: 0.0417
Epoch 64/300
6/6 [==============================] - 0s 4ms/step - loss: 3.4743 - accurac
y: 0.1302 - val_loss: 3.6769 - val_accuracy: 0.0208
Epoch 65/300
6/6 [==============================] - 0s 4ms/step - loss: 3.4730 - accurac
y: 0.1042 - val_loss: 3.6828 - val_accuracy: 0.0417
Epoch 66/300
6/6 [==============================] - 0s 4ms/step - loss: 3.4719 - accurac
y: 0.1094 - val_loss: 3.6875 - val_accuracy: 0.0208
Epoch 67/300
6/6 [==============================] - 0s 4ms/step - loss: 3.4643 - accurac
y: 0.1094 - val_loss: 3.6772 - val_accuracy: 0.0417
Epoch 68/300
6/6 [==============================] - 0s 4ms/step - loss: 3.4628 - accurac
y: 0.1302 - val_loss: 3.6771 - val_accuracy: 0.0208
Epoch 69/300
6/6 [==============================] - 0s 4ms/step - loss: 3.4589 - accurac
y: 0.1094 - val_loss: 3.6793 - val_accuracy: 0.0208
Epoch 70/300
6/6 [==============================] - 0s 4ms/step - loss: 3.4554 - accurac
y: 0.1146 - val_loss: 3.6700 - val_accuracy: 0.0208
Epoch 71/300
6/6 [==============================] - 0s 4ms/step - loss: 3.4523 - accurac
y: 0.1094 - val_loss: 3.6815 - val_accuracy: 0.0208
Epoch 72/300
6/6 [==============================] - 0s 4ms/step - loss: 3.4511 - accurac
y: 0.1042 - val_loss: 3.6674 - val_accuracy: 0.0208
Epoch 73/300
6/6 [==============================] - 0s 4ms/step - loss: 3.4474 - accurac
y: 0.1042 - val_loss: 3.6815 - val_accuracy: 0.0208
Epoch 74/300
6/6 [==============================] - 0s 4ms/step - loss: 3.4434 - accurac
y: 0.0938 - val_loss: 3.6676 - val_accuracy: 0.0208
Epoch 75/300
6/6 [==============================] - 0s 5ms/step - loss: 3.4441 - accurac
```

```
y: 0.1146 - val_loss: 3.6756 - val_accuracy: 0.0208
Epoch 76/300
6/6 [==============================] - 0s 4ms/step - loss: 3.4386 - accurac
y: 0.1198 - val_loss: 3.6696 - val_accuracy: 0.0208
Epoch 77/300
6/6 [==============================] - 0s 4ms/step - loss: 3.4414 - accurac
y: 0.1146 - val_loss: 3.6712 - val_accuracy: 0.0000e+00
Epoch 78/300
6/6 [==============================] - 0s 4ms/step - loss: 3.4346 - accurac
y: 0.1198 - val_loss: 3.6713 - val_accuracy: 0.0208
Epoch 79/300
6/6 [==============================] - 0s 4ms/step - loss: 3.4330 - accurac
y: 0.1042 - val_loss: 3.6635 - val_accuracy: 0.0000e+00
Epoch 80/300
6/6 [==============================] - 0s 4ms/step - loss: 3.4282 - accurac
y: 0.1042 - val_loss: 3.6824 - val_accuracy: 0.0208
Epoch 81/300
6/6 [==============================] - 0s 4ms/step - loss: 3.4327 - accurac
y: 0.1198 - val_loss: 3.6620 - val_accuracy: 0.0000e+00
Epoch 82/300
6/6 [==============================] - 0s 4ms/step - loss: 3.4228 - accurac
y: 0.1146 - val_loss: 3.6646 - val_accuracy: 0.0000e+00
Epoch 83/300
6/6 [==============================] - 0s 4ms/step - loss: 3.4204 - accurac
y: 0.1042 - val_loss: 3.6649 - val_accuracy: 0.0208
Epoch 84/300
6/6 [==============================] - 0s 4ms/step - loss: 3.4191 - accurac
y: 0.1146 - val_loss: 3.6651 - val_accuracy: 0.0000e+00
Epoch 85/300
6/6 [==============================] - 0s 4ms/step - loss: 3.4153 - accurac
y: 0.1042 - val_loss: 3.6621 - val_accuracy: 0.0000e+00
Epoch 86/300
6/6 [==============================] - 0s 4ms/step - loss: 3.4150 - accurac
y: 0.0938 - val_loss: 3.6633 - val_accuracy: 0.0000e+00
Epoch 87/300
6/6 [==============================] - 0s 4ms/step - loss: 3.4134 - accurac
y: 0.1250 - val_loss: 3.6631 - val_accuracy: 0.0000e+00
Epoch 88/300
6/6 [==============================] - 0s 4ms/step - loss: 3.4108 - accurac
y: 0.1094 - val_loss: 3.6686 - val_accuracy: 0.0208
Epoch 89/300
6/6 [==============================] - 0s 4ms/step - loss: 3.4060 - accurac
y: 0.1146 - val_loss: 3.6655 - val_accuracy: 0.0000e+00
Epoch 90/300
6/6 [==============================] - 0s 4ms/step - loss: 3.4045 - accurac
y: 0.1094 - val_loss: 3.6694 - val_accuracy: 0.0208
Epoch 91/300
6/6 [==============================] - 0s 4ms/step - loss: 3.3993 - accurac
y: 0.0990 - val_loss: 3.6570 - val_accuracy: 0.0000e+00
Epoch 92/300
6/6 [==============================] - 0s 4ms/step - loss: 3.3999 - accurac
y: 0.1042 - val_loss: 3.6598 - val_accuracy: 0.0000e+00
Epoch 93/300
6/6 [==============================] - 0s 4ms/step - loss: 3.3994 - accurac
y: 0.0938 - val_loss: 3.6529 - val_accuracy: 0.0000e+00
Epoch 94/300
```

```
6/6 [==============================] - 0s 4ms/step - loss: 3.3954 - accurac
y: 0.1042 - val_loss: 3.6605 - val_accuracy: 0.0000e+00
Epoch 95/300
6/6 [==============================] - 0s 4ms/step - loss: 3.3922 - accurac
y: 0.0938 - val_loss: 3.6513 - val_accuracy: 0.0000e+00
Epoch 96/300
6/6 [==============================] - 0s 4ms/step - loss: 3.3947 - accurac
y: 0.0990 - val_loss: 3.6674 - val_accuracy: 0.0000e+00
Epoch 97/300
6/6 [==============================] - 0s 4ms/step - loss: 3.3891 - accurac
y: 0.0938 - val_loss: 3.6476 - val_accuracy: 0.0000e+00
Epoch 98/300
6/6 [==============================] - 0s 4ms/step - loss: 3.3848 - accurac
y: 0.0990 - val_loss: 3.6668 - val_accuracy: 0.0000e+00
Epoch 99/300
6/6 [==============================] - 0s 4ms/step - loss: 3.3830 - accurac
y: 0.0990 - val_loss: 3.6464 - val_accuracy: 0.0000e+00
Epoch 100/300
6/6 [==============================] - 0s 4ms/step - loss: 3.3770 - accurac
y: 0.0885 - val_loss: 3.6546 - val_accuracy: 0.0000e+00
Epoch 101/300
6/6 [==============================] - 0s 4ms/step - loss: 3.3783 - accurac
y: 0.0990 - val_loss: 3.6521 - val_accuracy: 0.0000e+00
Epoch 102/300
6/6 [==============================] - 0s 4ms/step - loss: 3.3730 - accurac
y: 0.0938 - val_loss: 3.6440 - val_accuracy: 0.0000e+00
Epoch 103/300
6/6 [==============================] - 0s 4ms/step - loss: 3.3725 - accurac
y: 0.0938 - val_loss: 3.6482 - val_accuracy: 0.0000e+00
Epoch 104/300
6/6 [==============================] - 0s 4ms/step - loss: 3.3700 - accurac
y: 0.0938 - val_loss: 3.6515 - val_accuracy: 0.0000e+00
Epoch 105/300
6/6 [==============================] - 0s 4ms/step - loss: 3.3748 - accurac
y: 0.1042 - val_loss: 3.6607 - val_accuracy: 0.0000e+00
Epoch 106/300
6/6 [==============================] - 0s 4ms/step - loss: 3.3738 - accurac
y: 0.0990 - val_loss: 3.6644 - val_accuracy: 0.0000e+00
Epoch 107/300
6/6 [==============================] - 0s 4ms/step - loss: 3.3663 - accurac
y: 0.1146 - val_loss: 3.6197 - val_accuracy: 0.0625
Epoch 108/300
6/6 [==============================] - 0s 4ms/step - loss: 3.3651 - accurac
y: 0.1042 - val_loss: 3.6474 - val_accuracy: 0.0000e+00
Epoch 109/300
6/6 [==============================] - 0s 4ms/step - loss: 3.3681 - accurac
y: 0.0885 - val_loss: 3.6437 - val_accuracy: 0.0208
Epoch 110/300
6/6 [==============================] - 0s 4ms/step - loss: 3.3609 - accurac
y: 0.1042 - val_loss: 3.6443 - val_accuracy: 0.0000e+00
Epoch 111/300
6/6 [==============================] - 0s 4ms/step - loss: 3.3585 - accurac
y: 0.0938 - val_loss: 3.6455 - val_accuracy: 0.0208
Epoch 112/300
6/6 [==============================] - 0s 4ms/step - loss: 3.3521 - accurac
y: 0.0938 - val_loss: 3.6349 - val_accuracy: 0.0000e+00
```

```
Epoch 113/300
6/6 [==============================] - 0s 4ms/step - loss: 3.3464 - accurac
y: 0.1042 - val_loss: 3.6362 - val_accuracy: 0.0000e+00
Epoch 114/300
6/6 [==============================] - 0s 4ms/step - loss: 3.3469 - accurac
y: 0.1042 - val_loss: 3.6363 - val_accuracy: 0.0208
Epoch 115/300
6/6 [==============================] - 0s 4ms/step - loss: 3.3423 - accurac
y: 0.1042 - val_loss: 3.6397 - val_accuracy: 0.0000e+00
Epoch 116/300
6/6 [==============================] - 0s 4ms/step - loss: 3.3439 - accurac
y: 0.0938 - val_loss: 3.6296 - val_accuracy: 0.0000e+00
Epoch 117/300
6/6 [==============================] - 0s 4ms/step - loss: 3.3385 - accurac
y: 0.1094 - val_loss: 3.6400 - val_accuracy: 0.0000e+00
Epoch 118/300
6/6 [==============================] - 0s 4ms/step - loss: 3.3370 - accurac
y: 0.1042 - val_loss: 3.6265 - val_accuracy: 0.0208
Epoch 119/300
6/6 [==============================] - 0s 4ms/step - loss: 3.3370 - accurac
y: 0.1146 - val_loss: 3.6493 - val_accuracy: 0.0000e+00
Epoch 120/300
6/6 [==============================] - 0s 4ms/step - loss: 3.3403 - accurac
y: 0.1146 - val_loss: 3.6279 - val_accuracy: 0.0000e+00
Epoch 121/300
6/6 [==============================] - 0s 4ms/step - loss: 3.3308 - accurac
y: 0.0990 - val_loss: 3.6488 - val_accuracy: 0.0208
Epoch 122/300
6/6 [==============================] - 0s 4ms/step - loss: 3.3263 - accurac
y: 0.1146 - val_loss: 3.6240 - val_accuracy: 0.0000e+00
Epoch 123/300
6/6 [==============================] - 0s 4ms/step - loss: 3.3208 - accurac
y: 0.0990 - val_loss: 3.6292 - val_accuracy: 0.0000e+00
Epoch 124/300
6/6 [==============================] - 0s 4ms/step - loss: 3.3164 - accurac
y: 0.1198 - val_loss: 3.6178 - val_accuracy: 0.0000e+00
Epoch 125/300
6/6 [==============================] - 0s 4ms/step - loss: 3.3077 - accurac
y: 0.1094 - val_loss: 3.6180 - val_accuracy: 0.0000e+00
Epoch 126/300
6/6 [==============================] - 0s 4ms/step - loss: 3.3099 - accurac
y: 0.1146 - val_loss: 3.6243 - val_accuracy: 0.0000e+00
Epoch 127/300
6/6 [==============================] - 0s 5ms/step - loss: 3.3137 - accurac
y: 0.1146 - val_loss: 3.6199 - val_accuracy: 0.0417
Epoch 128/300
6/6 [==============================] - 0s 4ms/step - loss: 3.3114 - accurac
y: 0.1198 - val_loss: 3.6276 - val_accuracy: 0.0208
Epoch 129/300
6/6 [==============================] - 0s 4ms/step - loss: 3.3038 - accurac
y: 0.1354 - val_loss: 3.5991 - val_accuracy: 0.0000e+00
Epoch 130/300
6/6 [==============================] - 0s 4ms/step - loss: 3.2924 - accurac
y: 0.1354 - val_loss: 3.6207 - val_accuracy: 0.0208
Epoch 131/300
6/6 [==============================] - 0s 4ms/step - loss: 3.2917 - accurac
```

```
y: 0.0938 - val_loss: 3.5923 - val_accuracy: 0.0417
Epoch 132/300
6/6 [==============================] - 0s 4ms/step - loss: 3.2794 - accurac
y: 0.1302 - val_loss: 3.6098 - val_accuracy: 0.0417
Epoch 133/300
6/6 [==============================] - 0s 4ms/step - loss: 3.2759 - accurac
y: 0.1562 - val_loss: 3.5729 - val_accuracy: 0.0000e+00
Epoch 134/300
6/6 [==============================] - 0s 4ms/step - loss: 3.2631 - accurac
y: 0.1198 - val_loss: 3.5966 - val_accuracy: 0.0417
Epoch 135/300
6/6 [==============================] - 0s 4ms/step - loss: 3.2669 - accurac
y: 0.1042 - val_loss: 3.5782 - val_accuracy: 0.0417
Epoch 136/300
6/6 [==============================] - 0s 4ms/step - loss: 3.2524 - accurac
y: 0.1458 - val_loss: 3.5721 - val_accuracy: 0.0000e+00
Epoch 137/300
6/6 [==============================] - 0s 4ms/step - loss: 3.2506 - accurac
y: 0.1198 - val_loss: 3.5816 - val_accuracy: 0.0417
Epoch 138/300
6/6 [==============================] - 0s 4ms/step - loss: 3.2452 - accurac
y: 0.1354 - val_loss: 3.5577 - val_accuracy: 0.0208
Epoch 139/300
6/6 [==============================] - 0s 4ms/step - loss: 3.2343 - accurac
y: 0.1406 - val_loss: 3.5581 - val_accuracy: 0.0000e+00
Epoch 140/300
6/6 [==============================] - 0s 4ms/step - loss: 3.2339 - accurac
y: 0.1250 - val_loss: 3.5612 - val_accuracy: 0.0417
Epoch 141/300
6/6 [==============================] - 0s 4ms/step - loss: 3.2273 - accurac
y: 0.0990 - val_loss: 3.5554 - val_accuracy: 0.0417
Epoch 142/300
6/6 [==============================] - 0s 4ms/step - loss: 3.2227 - accurac
y: 0.1562 - val_loss: 3.5495 - val_accuracy: 0.0000e+00
Epoch 143/300
6/6 [==============================] - 0s 4ms/step - loss: 3.2153 - accurac
y: 0.1458 - val_loss: 3.5444 - val_accuracy: 0.0417
Epoch 144/300
6/6 [==============================] - 0s 4ms/step - loss: 3.2138 - accurac
y: 0.0990 - val_loss: 3.5577 - val_accuracy: 0.0417
Epoch 145/300
6/6 [==============================] - 0s 4ms/step - loss: 3.2378 - accurac
y: 0.1979 - val_loss: 3.5521 - val_accuracy: 0.0417
Epoch 146/300
6/6 [==============================] - 0s 4ms/step - loss: 3.2070 - accurac
y: 0.1302 - val_loss: 3.5385 - val_accuracy: 0.0625
Epoch 147/300
6/6 [==============================] - 0s 4ms/step - loss: 3.2044 - accurac
y: 0.1667 - val_loss: 3.5374 - val_accuracy: 0.0000e+00
Epoch 148/300
6/6 [==============================] - 0s 4ms/step - loss: 3.1989 - accurac
y: 0.1562 - val_loss: 3.5279 - val_accuracy: 0.0625
Epoch 149/300
6/6 [==============================] - 0s 4ms/step - loss: 3.1877 - accurac
y: 0.1302 - val_loss: 3.5332 - val_accuracy: 0.0625
Epoch 150/300
```

```
6/6 [==============================] - 0s 4ms/step - loss: 3.1830 - accurac
y: 0.1771 - val_loss: 3.5160 - val_accuracy: 0.0625
Epoch 151/300
6/6 [==============================] - 0s 4ms/step - loss: 3.1796 - accurac
y: 0.1406 - val_loss: 3.5348 - val_accuracy: 0.0625
Epoch 152/300
6/6 [==============================] - 0s 4ms/step - loss: 3.1742 - accurac
y: 0.1458 - val_loss: 3.5148 - val_accuracy: 0.0208
Epoch 153/300
6/6 [==============================] - 0s 4ms/step - loss: 3.1670 - accurac
y: 0.1510 - val_loss: 3.5183 - val_accuracy: 0.0625
Epoch 154/300
6/6 [==============================] - 0s 5ms/step - loss: 3.1632 - accurac
y: 0.1667 - val_loss: 3.5140 - val_accuracy: 0.0208
Epoch 155/300
6/6 [==============================] - 0s 4ms/step - loss: 3.1633 - accurac
y: 0.1719 - val_loss: 3.5079 - val_accuracy: 0.0625
Epoch 156/300
6/6 [==============================] - 0s 4ms/step - loss: 3.1568 - accurac
y: 0.1406 - val_loss: 3.5021 - val_accuracy: 0.0625
Epoch 157/300
6/6 [==============================] - 0s 4ms/step - loss: 3.1514 - accurac
y: 0.1979 - val_loss: 3.5020 - val_accuracy: 0.0208
Epoch 158/300
6/6 [==============================] - 0s 4ms/step - loss: 3.1495 - accurac
y: 0.1615 - val_loss: 3.5048 - val_accuracy: 0.0625
Epoch 159/300
6/6 [==============================] - 0s 4ms/step - loss: 3.1411 - accurac
y: 0.1875 - val_loss: 3.5124 - val_accuracy: 0.0208
Epoch 160/300
6/6 [==============================] - 0s 4ms/step - loss: 3.1424 - accurac
y: 0.1927 - val_loss: 3.4969 - val_accuracy: 0.0625
Epoch 161/300
6/6 [==============================] - 0s 4ms/step - loss: 3.1368 - accurac
y: 0.1823 - val_loss: 3.4884 - val_accuracy: 0.0625
Epoch 162/300
6/6 [==============================] - 0s 4ms/step - loss: 3.1295 - accurac
y: 0.1823 - val_loss: 3.4942 - val_accuracy: 0.0208
Epoch 163/300
6/6 [==============================] - 0s 4ms/step - loss: 3.1269 - accurac
y: 0.1719 - val_loss: 3.4890 - val_accuracy: 0.0625
Epoch 164/300
6/6 [==============================] - 0s 5ms/step - loss: 3.1204 - accurac
y: 0.1927 - val_loss: 3.4868 - val_accuracy: 0.0208
Epoch 165/300
6/6 [==============================] - 0s 4ms/step - loss: 3.1184 - accurac
y: 0.1719 - val_loss: 3.4758 - val_accuracy: 0.0625
Epoch 166/300
6/6 [==============================] - 0s 4ms/step - loss: 3.1160 - accurac
y: 0.1823 - val_loss: 3.4882 - val_accuracy: 0.0208
Epoch 167/300
6/6 [==============================] - 0s 5ms/step - loss: 3.1124 - accurac
y: 0.1979 - val_loss: 3.4711 - val_accuracy: 0.0625
Epoch 168/300
6/6 [==============================] - 0s 4ms/step - loss: 3.1145 - accurac
y: 0.1667 - val_loss: 3.4939 - val_accuracy: 0.0625
```

```
Epoch 169/300
6/6 [==============================] - 0s 4ms/step - loss: 3.1111 - accurac
y: 0.2031 - val_loss: 3.4797 - val_accuracy: 0.0208
Epoch 170/300
6/6 [==============================] - 0s 4ms/step - loss: 3.1071 - accurac
y: 0.1771 - val_loss: 3.4748 - val_accuracy: 0.0833
Epoch 171/300
6/6 [==============================] - 0s 5ms/step - loss: 3.1024 - accurac
y: 0.1771 - val_loss: 3.4764 - val_accuracy: 0.0417
Epoch 172/300
6/6 [==============================] - 0s 4ms/step - loss: 3.0898 - accurac
y: 0.1875 - val_loss: 3.4599 - val_accuracy: 0.0208
Epoch 173/300
6/6 [==============================] - 0s 4ms/step - loss: 3.0866 - accurac
y: 0.1719 - val_loss: 3.4731 - val_accuracy: 0.0417
Epoch 174/300
6/6 [==============================] - 0s 4ms/step - loss: 3.0829 - accurac
y: 0.1927 - val_loss: 3.4636 - val_accuracy: 0.0417
Epoch 175/300
6/6 [==============================] - 0s 4ms/step - loss: 3.0819 - accurac
y: 0.1823 - val_loss: 3.4622 - val_accuracy: 0.0625
Epoch 176/300
6/6 [==============================] - 0s 4ms/step - loss: 3.0742 - accurac
y: 0.1771 - val_loss: 3.4507 - val_accuracy: 0.0208
Epoch 177/300
6/6 [==============================] - 0s 4ms/step - loss: 3.0713 - accurac
y: 0.1927 - val_loss: 3.4700 - val_accuracy: 0.0417
Epoch 178/300
6/6 [==============================] - 0s 4ms/step - loss: 3.0691 - accurac
y: 0.1823 - val_loss: 3.4445 - val_accuracy: 0.0417
Epoch 179/300
6/6 [==============================] - 0s 4ms/step - loss: 3.0642 - accurac
y: 0.1771 - val_loss: 3.4460 - val_accuracy: 0.0417
Epoch 180/300
6/6 [==============================] - 0s 4ms/step - loss: 3.0604 - accurac
y: 0.1875 - val_loss: 3.4438 - val_accuracy: 0.0208
Epoch 181/300
6/6 [==============================] - 0s 4ms/step - loss: 3.0543 - accurac
y: 0.1823 - val_loss: 3.4362 - val_accuracy: 0.0417
Epoch 182/300
6/6 [==============================] - 0s 5ms/step - loss: 3.0535 - accurac
y: 0.1979 - val_loss: 3.4451 - val_accuracy: 0.0208
Epoch 183/300
6/6 [==============================] - 0s 4ms/step - loss: 3.0477 - accurac
y: 0.1927 - val_loss: 3.4375 - val_accuracy: 0.0417
Epoch 184/300
6/6 [==============================] - 0s 4ms/step - loss: 3.0439 - accurac
y: 0.1771 - val_loss: 3.4281 - val_accuracy: 0.0417
Epoch 185/300
6/6 [==============================] - 0s 4ms/step - loss: 3.0411 - accurac
y: 0.1875 - val_loss: 3.4465 - val_accuracy: 0.0417
Epoch 186/300
6/6 [==============================] - 0s 4ms/step - loss: 3.0409 - accurac
y: 0.1771 - val_loss: 3.4290 - val_accuracy: 0.0625
Epoch 187/300
6/6 [==============================] - 0s 4ms/step - loss: 3.0443 - accurac
```

```
y: 0.1979 - val_loss: 3.4406 - val_accuracy: 0.0208
Epoch 188/300
6/6 [==============================] - 0s 4ms/step - loss: 3.0432 - accurac
y: 0.2135 - val_loss: 3.4460 - val_accuracy: 0.0833
Epoch 189/300
6/6 [==============================] - 0s 5ms/step - loss: 3.0363 - accurac
y: 0.1823 - val_loss: 3.4249 - val_accuracy: 0.0208
Epoch 190/300
6/6 [==============================] - 0s 4ms/step - loss: 3.0320 - accurac
y: 0.1927 - val_loss: 3.4278 - val_accuracy: 0.0417
Epoch 191/300
6/6 [==============================] - 0s 4ms/step - loss: 3.0239 - accurac
y: 0.1823 - val_loss: 3.4197 - val_accuracy: 0.0417
Epoch 192/300
6/6 [==============================] - 0s 4ms/step - loss: 3.0157 - accurac
y: 0.1979 - val_loss: 3.4249 - val_accuracy: 0.0208
Epoch 193/300
6/6 [==============================] - 0s 4ms/step - loss: 3.0118 - accurac
y: 0.1875 - val_loss: 3.4114 - val_accuracy: 0.0417
Epoch 194/300
6/6 [==============================] - 0s 4ms/step - loss: 3.0103 - accurac
y: 0.1927 - val_loss: 3.4121 - val_accuracy: 0.0208
Epoch 195/300
6/6 [==============================] - 0s 4ms/step - loss: 3.0088 - accurac
y: 0.1823 - val_loss: 3.4173 - val_accuracy: 0.0417
Epoch 196/300
6/6 [==============================] - 0s 4ms/step - loss: 3.0066 - accurac
y: 0.1979 - val_loss: 3.4172 - val_accuracy: 0.0417
Epoch 197/300
6/6 [==============================] - 0s 4ms/step - loss: 3.0035 - accurac
y: 0.1875 - val_loss: 3.4155 - val_accuracy: 0.0208
Epoch 198/300
6/6 [==============================] - 0s 4ms/step - loss: 2.9998 - accurac
y: 0.1875 - val_loss: 3.4129 - val_accuracy: 0.0417
Epoch 199/300
6/6 [==============================] - 0s 4ms/step - loss: 2.9920 - accurac
y: 0.2031 - val_loss: 3.4054 - val_accuracy: 0.0208
Epoch 200/300
6/6 [==============================] - 0s 4ms/step - loss: 2.9891 - accurac
y: 0.1823 - val_loss: 3.3992 - val_accuracy: 0.0208
Epoch 201/300
6/6 [==============================] - 0s 5ms/step - loss: 2.9845 - accurac
y: 0.1875 - val_loss: 3.3975 - val_accuracy: 0.0417
Epoch 202/300
6/6 [==============================] - 0s 4ms/step - loss: 2.9860 - accurac
y: 0.1927 - val_loss: 3.4112 - val_accuracy: 0.0208
Epoch 203/300
6/6 [==============================] - 0s 4ms/step - loss: 2.9799 - accurac
y: 0.1823 - val_loss: 3.3858 - val_accuracy: 0.0417
Epoch 204/300
6/6 [==============================] - 0s 4ms/step - loss: 2.9745 - accurac
y: 0.1875 - val_loss: 3.4000 - val_accuracy: 0.0417
Epoch 205/300
6/6 [==============================] - 0s 4ms/step - loss: 2.9706 - accurac
y: 0.1927 - val_loss: 3.3871 - val_accuracy: 0.0417
Epoch 206/300
```

```
6/6 [==============================] - 0s 4ms/step - loss: 2.9677 - accurac
y: 0.1823 - val_loss: 3.3983 - val_accuracy: 0.0417
Epoch 207/300
6/6 [==============================] - 0s 4ms/step - loss: 2.9660 - accurac
y: 0.1875 - val_loss: 3.3874 - val_accuracy: 0.0417
Epoch 208/300
6/6 [==============================] - 0s 4ms/step - loss: 2.9615 - accurac
y: 0.1875 - val_loss: 3.3753 - val_accuracy: 0.0417
Epoch 209/300
6/6 [==============================] - 0s 4ms/step - loss: 2.9603 - accurac
y: 0.1875 - val_loss: 3.3882 - val_accuracy: 0.0417
Epoch 210/300
6/6 [==============================] - 0s 4ms/step - loss: 2.9549 - accurac
y: 0.1823 - val_loss: 3.3779 - val_accuracy: 0.0417
Epoch 211/300
6/6 [==============================] - 0s 4ms/step - loss: 2.9505 - accurac
y: 0.1823 - val_loss: 3.3827 - val_accuracy: 0.0417
Epoch 212/300
6/6 [==============================] - 0s 4ms/step - loss: 2.9472 - accurac
y: 0.1979 - val_loss: 3.3718 - val_accuracy: 0.0417
Epoch 213/300
6/6 [==============================] - 0s 4ms/step - loss: 2.9444 - accurac
y: 0.1823 - val_loss: 3.3769 - val_accuracy: 0.0417
Epoch 214/300
6/6 [==============================] - 0s 4ms/step - loss: 2.9415 - accurac
y: 0.1771 - val_loss: 3.3623 - val_accuracy: 0.0417
Epoch 215/300
6/6 [==============================] - 0s 4ms/step - loss: 2.9399 - accurac
y: 0.1719 - val_loss: 3.3808 - val_accuracy: 0.0417
Epoch 216/300
6/6 [==============================] - 0s 4ms/step - loss: 2.9359 - accurac
y: 0.1875 - val_loss: 3.3589 - val_accuracy: 0.0417
Epoch 217/300
6/6 [==============================] - 0s 4ms/step - loss: 2.9337 - accurac
y: 0.1771 - val_loss: 3.3632 - val_accuracy: 0.0417
Epoch 218/300
6/6 [==============================] - 0s 4ms/step - loss: 2.9305 - accurac
y: 0.1667 - val_loss: 3.3535 - val_accuracy: 0.0417
Epoch 219/300
6/6 [==============================] - 0s 4ms/step - loss: 2.9250 - accurac
y: 0.1823 - val_loss: 3.3677 - val_accuracy: 0.0417
Epoch 220/300
6/6 [==============================] - 0s 5ms/step - loss: 2.9217 - accurac
y: 0.1771 - val_loss: 3.3498 - val_accuracy: 0.0208
Epoch 221/300
6/6 [==============================] - 0s 4ms/step - loss: 2.9172 - accurac
y: 0.1823 - val_loss: 3.3696 - val_accuracy: 0.0417
Epoch 222/300
6/6 [==============================] - 0s 4ms/step - loss: 2.9150 - accurac
y: 0.1771 - val_loss: 3.3430 - val_accuracy: 0.0417
Epoch 223/300
6/6 [==============================] - 0s 4ms/step - loss: 2.9137 - accurac
y: 0.1823 - val_loss: 3.3645 - val_accuracy: 0.0417
Epoch 224/300
6/6 [==============================] - 0s 4ms/step - loss: 2.9135 - accurac
y: 0.1875 - val_loss: 3.3431 - val_accuracy: 0.0417
```

```
Epoch 225/300
6/6 [==============================] - 0s 4ms/step - loss: 2.9067 - accurac
y: 0.2031 - val_loss: 3.3519 - val_accuracy: 0.0417
Epoch 226/300
6/6 [==============================] - 0s 4ms/step - loss: 2.9030 - accurac
y: 0.1927 - val_loss: 3.3341 - val_accuracy: 0.0417
Epoch 227/300
6/6 [==============================] - 0s 4ms/step - loss: 2.9005 - accurac
y: 0.1927 - val_loss: 3.3493 - val_accuracy: 0.0417
Epoch 228/300
6/6 [==============================] - 0s 5ms/step - loss: 2.8986 - accurac
y: 0.1927 - val_loss: 3.3466 - val_accuracy: 0.0417
Epoch 229/300
6/6 [==============================] - 0s 4ms/step - loss: 2.8989 - accurac
y: 0.1875 - val_loss: 3.3327 - val_accuracy: 0.0417
Epoch 230/300
6/6 [==============================] - 0s 4ms/step - loss: 2.8974 - accurac
y: 0.1979 - val_loss: 3.3680 - val_accuracy: 0.0417
Epoch 231/300
6/6 [==============================] - 0s 4ms/step - loss: 2.8962 - accurac
y: 0.1979 - val_loss: 3.3221 - val_accuracy: 0.0417
Epoch 232/300
6/6 [==============================] - 0s 4ms/step - loss: 2.8878 - accurac
y: 0.2031 - val_loss: 3.3451 - val_accuracy: 0.0208
Epoch 233/300
6/6 [==============================] - 0s 4ms/step - loss: 2.8843 - accurac
y: 0.1927 - val_loss: 3.3253 - val_accuracy: 0.0417
Epoch 234/300
6/6 [==============================] - 0s 4ms/step - loss: 2.8802 - accurac
y: 0.1927 - val_loss: 3.3226 - val_accuracy: 0.0208
Epoch 235/300
6/6 [==============================] - 0s 4ms/step - loss: 2.8788 - accurac
y: 0.2031 - val_loss: 3.3300 - val_accuracy: 0.0417
Epoch 236/300
6/6 [==============================] - 0s 4ms/step - loss: 2.8780 - accurac
y: 0.1927 - val_loss: 3.3248 - val_accuracy: 0.0417
Epoch 237/300
6/6 [==============================] - 0s 4ms/step - loss: 2.8760 - accurac
y: 0.1979 - val_loss: 3.3140 - val_accuracy: 0.0417
Epoch 238/300
6/6 [==============================] - 0s 4ms/step - loss: 2.8673 - accurac
y: 0.1927 - val_loss: 3.3141 - val_accuracy: 0.0417
Epoch 239/300
6/6 [==============================] - 0s 4ms/step - loss: 2.8612 - accurac
y: 0.2083 - val_loss: 3.3453 - val_accuracy: 0.0417
Epoch 240/300
6/6 [==============================] - 0s 5ms/step - loss: 2.8603 - accurac
y: 0.1875 - val_loss: 3.3146 - val_accuracy: 0.0417
Epoch 241/300
6/6 [==============================] - 0s 4ms/step - loss: 2.8593 - accurac
y: 0.2188 - val_loss: 3.3221 - val_accuracy: 0.0208
Epoch 242/300
6/6 [==============================] - 0s 4ms/step - loss: 2.8524 - accurac
y: 0.2135 - val_loss: 3.3003 - val_accuracy: 0.0417
Epoch 243/300
6/6 [==============================] - 0s 4ms/step - loss: 2.8486 - accurac
```

```
y: 0.1979 - val_loss: 3.3237 - val_accuracy: 0.0417
Epoch 244/300
6/6 [==============================] - 0s 4ms/step - loss: 2.8528 - accurac
y: 0.2083 - val_loss: 3.3061 - val_accuracy: 0.0208
Epoch 245/300
6/6 [==============================] - 0s 4ms/step - loss: 2.8344 - accurac
y: 0.2031 - val_loss: 3.3052 - val_accuracy: 0.0417
Epoch 246/300
6/6 [==============================] - 0s 4ms/step - loss: 2.8282 - accurac
y: 0.1979 - val_loss: 3.3214 - val_accuracy: 0.0208
Epoch 247/300
6/6 [==============================] - 0s 4ms/step - loss: 2.8443 - accurac
y: 0.1927 - val_loss: 3.2878 - val_accuracy: 0.0417
Epoch 248/300
6/6 [==============================] - 0s 4ms/step - loss: 2.8407 - accurac
y: 0.2031 - val_loss: 3.3144 - val_accuracy: 0.0417
Epoch 249/300
6/6 [==============================] - 0s 4ms/step - loss: 2.8415 - accurac
y: 0.2031 - val_loss: 3.2890 - val_accuracy: 0.0417
Epoch 250/300
6/6 [==============================] - 0s 4ms/step - loss: 2.8364 - accurac
y: 0.1927 - val_loss: 3.2876 - val_accuracy: 0.0417
Epoch 251/300
6/6 [==============================] - 0s 4ms/step - loss: 2.8326 - accurac
y: 0.1927 - val_loss: 3.3058 - val_accuracy: 0.0417
Epoch 252/300
6/6 [==============================] - 0s 4ms/step - loss: 2.8310 - accurac
y: 0.1979 - val_loss: 3.2759 - val_accuracy: 0.0417
Epoch 253/300
6/6 [==============================] - 0s 4ms/step - loss: 2.8316 - accurac
y: 0.2135 - val_loss: 3.3130 - val_accuracy: 0.0417
Epoch 254/300
6/6 [==============================] - 0s 4ms/step - loss: 2.8239 - accurac
y: 0.1979 - val_loss: 3.2748 - val_accuracy: 0.0417
Epoch 255/300
6/6 [==============================] - 0s 4ms/step - loss: 2.8233 - accurac
y: 0.1927 - val_loss: 3.2870 - val_accuracy: 0.0417
Epoch 256/300
6/6 [==============================] - 0s 4ms/step - loss: 2.8169 - accurac
y: 0.2083 - val_loss: 3.2764 - val_accuracy: 0.0417
Epoch 257/300
6/6 [==============================] - 0s 4ms/step - loss: 2.8147 - accurac
y: 0.1979 - val_loss: 3.2811 - val_accuracy: 0.0417
Epoch 258/300
6/6 [==============================] - 0s 4ms/step - loss: 2.8129 - accurac
y: 0.2031 - val_loss: 3.2703 - val_accuracy: 0.0208
Epoch 259/300
6/6 [==============================] - 0s 4ms/step - loss: 2.8141 - accurac
y: 0.2135 - val_loss: 3.2748 - val_accuracy: 0.0417
Epoch 260/300
6/6 [==============================] - 0s 4ms/step - loss: 2.8078 - accurac
y: 0.1979 - val_loss: 3.2858 - val_accuracy: 0.0417
Epoch 261/300
6/6 [==============================] - 0s 5ms/step - loss: 2.8058 - accurac
y: 0.1979 - val_loss: 3.2634 - val_accuracy: 0.0417
Epoch 262/300
```

```
6/6 [==============================] - 0s 4ms/step - loss: 2.8058 - accurac
y: 0.2083 - val_loss: 3.2937 - val_accuracy: 0.0417
Epoch 263/300
6/6 [==============================] - 0s 4ms/step - loss: 2.7994 - accurac
y: 0.1979 - val_loss: 3.2531 - val_accuracy: 0.0417
Epoch 264/300
6/6 [==============================] - 0s 4ms/step - loss: 2.7986 - accurac
y: 0.2083 - val_loss: 3.3021 - val_accuracy: 0.0417
Epoch 265/300
6/6 [==============================] - 0s 5ms/step - loss: 2.7942 - accurac
y: 0.1979 - val_loss: 3.2494 - val_accuracy: 0.0417
Epoch 266/300
6/6 [==============================] - 0s 4ms/step - loss: 2.7940 - accurac
y: 0.2083 - val_loss: 3.2615 - val_accuracy: 0.0417
Epoch 267/300
6/6 [==============================] - 0s 4ms/step - loss: 2.7906 - accurac
y: 0.2031 - val_loss: 3.2766 - val_accuracy: 0.0417
Epoch 268/300
6/6 [==============================] - 0s 4ms/step - loss: 2.7879 - accurac
y: 0.2031 - val_loss: 3.2567 - val_accuracy: 0.0208
Epoch 269/300
6/6 [==============================] - 0s 4ms/step - loss: 2.7834 - accurac
y: 0.2031 - val_loss: 3.2542 - val_accuracy: 0.0417
Epoch 270/300
6/6 [==============================] - 0s 4ms/step - loss: 2.7786 - accurac
y: 0.2031 - val_loss: 3.2608 - val_accuracy: 0.0417
Epoch 271/300
6/6 [==============================] - 0s 5ms/step - loss: 2.7784 - accurac
y: 0.2083 - val_loss: 3.2419 - val_accuracy: 0.0417
Epoch 272/300
6/6 [==============================] - 0s 4ms/step - loss: 2.7722 - accurac
y: 0.2083 - val_loss: 3.2677 - val_accuracy: 0.0417
Epoch 273/300
6/6 [==============================] - 0s 4ms/step - loss: 2.7708 - accurac
y: 0.2135 - val_loss: 3.2423 - val_accuracy: 0.0417
Epoch 274/300
6/6 [==============================] - 0s 4ms/step - loss: 2.7689 - accurac
y: 0.2188 - val_loss: 3.2559 - val_accuracy: 0.0417
Epoch 275/300
6/6 [==============================] - 0s 4ms/step - loss: 2.7659 - accurac
y: 0.2031 - val_loss: 3.2421 - val_accuracy: 0.0417
Epoch 276/300
6/6 [==============================] - 0s 4ms/step - loss: 2.7653 - accurac
y: 0.2083 - val_loss: 3.2332 - val_accuracy: 0.0417
Epoch 277/300
6/6 [==============================] - 0s 4ms/step - loss: 2.7632 - accurac
y: 0.2135 - val_loss: 3.2599 - val_accuracy: 0.0417
Epoch 278/300
6/6 [==============================] - 0s 4ms/step - loss: 2.7573 - accurac
y: 0.2135 - val_loss: 3.2259 - val_accuracy: 0.0417
Epoch 279/300
6/6 [==============================] - 0s 4ms/step - loss: 2.7563 - accurac
y: 0.2083 - val_loss: 3.2659 - val_accuracy: 0.0417
Epoch 280/300
6/6 [==============================] - 0s 4ms/step - loss: 2.7552 - accurac
y: 0.2135 - val_loss: 3.2263 - val_accuracy: 0.0417
```

```
Epoch 281/300
6/6 [==============================] - 0s 4ms/step - loss: 2.7519 - accurac
y: 0.2083 - val_loss: 3.2579 - val_accuracy: 0.0625
Epoch 282/300
6/6 [==============================] - 0s 4ms/step - loss: 2.7530 - accurac
y: 0.2135 - val_loss: 3.2212 - val_accuracy: 0.0208
Epoch 283/300
6/6 [==============================] - 0s 4ms/step - loss: 2.7473 - accurac
y: 0.2188 - val_loss: 3.2497 - val_accuracy: 0.0625
Epoch 284/300
6/6 [==============================] - 0s 4ms/step - loss: 2.7481 - accurac
y: 0.2188 - val_loss: 3.2226 - val_accuracy: 0.0417
Epoch 285/300
6/6 [==============================] - 0s 5ms/step - loss: 2.7408 - accurac
y: 0.2188 - val_loss: 3.2299 - val_accuracy: 0.0625
Epoch 286/300
6/6 [==============================] - 0s 4ms/step - loss: 2.7377 - accurac
y: 0.2188 - val_loss: 3.2278 - val_accuracy: 0.0417
Epoch 287/300
6/6 [==============================] - 0s 4ms/step - loss: 2.7360 - accurac
y: 0.2135 - val_loss: 3.2170 - val_accuracy: 0.0625
Epoch 288/300
6/6 [==============================] - 0s 5ms/step - loss: 2.7356 - accurac
y: 0.2135 - val_loss: 3.2447 - val_accuracy: 0.0625
Epoch 289/300
6/6 [==============================] - 0s 4ms/step - loss: 2.7329 - accurac
y: 0.2188 - val_loss: 3.2039 - val_accuracy: 0.0208
Epoch 290/300
6/6 [==============================] - 0s 5ms/step - loss: 2.7302 - accurac
y: 0.2135 - val_loss: 3.2337 - val_accuracy: 0.0625
Epoch 291/300
6/6 [==============================] - 0s 4ms/step - loss: 2.7274 - accurac
y: 0.2083 - val_loss: 3.2055 - val_accuracy: 0.0417
Epoch 292/300
6/6 [==============================] - 0s 4ms/step - loss: 2.7256 - accurac
y: 0.2188 - val_loss: 3.2105 - val_accuracy: 0.0625
Epoch 293/300
6/6 [==============================] - 0s 4ms/step - loss: 2.7227 - accurac
y: 0.2188 - val_loss: 3.2143 - val_accuracy: 0.0417
Epoch 294/300
6/6 [==============================] - 0s 4ms/step - loss: 2.7177 - accurac
y: 0.2135 - val_loss: 3.2048 - val_accuracy: 0.0625
Epoch 295/300
6/6 [==============================] - 0s 4ms/step - loss: 2.7160 - accurac
y: 0.2135 - val_loss: 3.2194 - val_accuracy: 0.0625
Epoch 296/300
6/6 [==============================] - 0s 4ms/step - loss: 2.7126 - accurac
y: 0.2135 - val_loss: 3.1958 - val_accuracy: 0.0417
Epoch 297/300
6/6 [==============================] - 0s 4ms/step - loss: 2.7118 - accurac
y: 0.2188 - val_loss: 3.2199 - val_accuracy: 0.0625
Epoch 298/300
6/6 [==============================] - 0s 4ms/step - loss: 2.7105 - accurac
y: 0.2135 - val_loss: 3.1974 - val_accuracy: 0.0417
Epoch 299/300
6/6 [==============================] - 0s 4ms/step - loss: 2.7081 - accurac
```

```
y: 0.2188 - val_loss: 3.1979 - val_accuracy: 0.0417
Epoch 300/300
6/6 [==============================] - 0s 4ms/step - loss: 2.7038 - accurac
y: 0.2031 - val_loss: 3.2012 - val_accuracy: 0.0417
```

In [9]:
```
test_loss, test_acc = model.evaluate(x_test,  y_test, verbose=2)

print('\nTest accuracy:', test_acc)
```

```
5/5 - 0s - loss: 3.0558 - accuracy: 0.1000 - 14ms/epoch - 3ms/step

Test accuracy: 0.10000000149011612
```

Please use one time 10-fold cross validation to compare the performance of different neural network architectures, including (1) one-hidden layer NN with 10 hidden nodes, (2) one-hidden layer NN with 50 hidden nodes, (3) one-hidden layer NN with 500 hidden nodes, and (4) two-hidden layer NN with 50 hidden nodes (1st layer) and 10 hidden nodes (2n layer). Please report and compare the cross-validation accuracy of the four neural networks, and conclude which classifier has the best performance [1 pt].

> The accuracy is lowest for the two architectures. The number of nodes in the hidden layer is not high enough to capture the complex structure of the data. The higher number of nodes in the 3rd architecture is able to do so and thus give the highest accuracy. Even though the 4th architecture has two hidden layers, the lower number of nodes in the 2nd hidden layer is not sufficient for this problem. So, the 3rd architecture performs the best.

In [10]:
```
### one-hidden layer NN with 10 hidden nodes,

kfold = StratifiedKFold(n_splits=10, shuffle=True)
accuracy = []


for train_index, test_index in kfold.split(x,y):
    model = keras.Sequential([
    keras.layers.Flatten(input_shape=(64, 64, 1),name="Input"),
    keras.layers.Dense(10, activation='sigmoid',name="Hidden"),
    keras.layers.Dense(40, name="Output"),
    ])

    model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits
              metrics=['accuracy'])

    model.fit(x_train, y_train, epochs=300, batch_size=32, verbose = 0)
    scores = model.evaluate(x_test, y_test)
    accuracy.append(scores[1] * 100)


print("Accuracy: %.2f%%" % (np.mean(accuracy)))
```

```
5/5 [==============================] - 0s 1ms/step - loss: 3.1910 - accurac
y: 0.0688
5/5 [==============================] - 0s 1ms/step - loss: 3.1424 - accurac
y: 0.0562
5/5 [==============================] - 0s 1ms/step - loss: 2.8020 - accurac
y: 0.1375
5/5 [==============================] - 0s 1ms/step - loss: 2.4891 - accurac
y: 0.2250
5/5 [==============================] - 0s 1ms/step - loss: 3.2033 - accurac
y: 0.0812
5/5 [==============================] - 0s 1ms/step - loss: 2.4654 - accurac
y: 0.2250
5/5 [==============================] - 0s 966us/step - loss: 2.7519 - accura
cy: 0.1688
5/5 [==============================] - 0s 1ms/step - loss: 3.0543 - accurac
y: 0.0625
5/5 [==============================] - 0s 1ms/step - loss: 3.4961 - accurac
y: 0.0250
5/5 [==============================] - 0s 1ms/step - loss: 2.4642 - accurac
y: 0.3063
Accuracy: 13.56%
```

In [11]:

```
###  (2) one-hidden layer NN with 50 hidden nodes

accuracy = []


for train_index, test_index in kfold.split(x,y):
    model = keras.Sequential([
    keras.layers.Flatten(input_shape=(64, 64, 1),name="Input"),
    keras.layers.Dense(10, activation='sigmoid',name="Hidden"),
    keras.layers.Dense(50, name="Output"),
    ])

    model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits
              metrics=['accuracy'])

    model.fit(x_train, y_train, epochs=300, batch_size=32, verbose = 0)
    scores = model.evaluate(x_test, y_test)
    accuracy.append(scores[1] * 100)


print("Accuracy: %.2f%%" % (np.mean(accuracy)))
```

```
5/5 [==============================] - 0s 926us/step - loss: 2.6127 - accura
cy: 0.2375
5/5 [==============================] - 0s 1ms/step - loss: 3.3362 - accurac
y: 0.0750
5/5 [==============================] - 0s 1ms/step - loss: 3.2282 - accurac
y: 0.0688
5/5 [==============================] - 0s 1ms/step - loss: 2.6217 - accurac
y: 0.2937
5/5 [==============================] - 0s 1ms/step - loss: 2.7150 - accurac
y: 0.2125
5/5 [==============================] - 0s 1ms/step - loss: 2.2834 - accurac
y: 0.4000
5/5 [==============================] - 0s 1ms/step - loss: 2.8716 - accurac
y: 0.1250
5/5 [==============================] - 0s 1ms/step - loss: 3.1812 - accurac
y: 0.0562
5/5 [==============================] - 0s 1ms/step - loss: 2.8569 - accurac
y: 0.1063
5/5 [==============================] - 0s 1ms/step - loss: 2.5394 - accurac
y: 0.2875
Accuracy: 18.62%
```

In [12]:

```
###  (3) one-hidden layer NN with 500 hidden nodes

kfold = StratifiedKFold(n_splits=10, shuffle=True)
accuracy = []


for train_index, test_index in kfold.split(x,y):
    model = keras.Sequential([
    keras.layers.Flatten(input_shape=(64, 64, 1),name="Input"),
    keras.layers.Dense(500, activation='sigmoid',name="Hidden"),
    keras.layers.Dense(40, name="Output"),
    ])

    model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits
              metrics=['accuracy'])

    model.fit(x_train, y_train, epochs=300, batch_size=32, verbose = 0)
    scores = model.evaluate(x_test, y_test)
    accuracy.append(scores[1] * 100)


print("Accuracy: %.2f%%" % (np.mean(accuracy)))
```

```
5/5 [==============================] - 0s 2ms/step - loss: 0.1683 - accurac
y: 0.9500
5/5 [==============================] - 0s 2ms/step - loss: 0.1778 - accurac
y: 0.9500
5/5 [==============================] - 0s 2ms/step - loss: 0.1909 - accurac
y: 0.9563
5/5 [==============================] - 0s 2ms/step - loss: 0.1776 - accurac
y: 0.9500
5/5 [==============================] - 0s 2ms/step - loss: 0.1708 - accurac
y: 0.9500
5/5 [==============================] - 0s 2ms/step - loss: 0.1658 - accurac
y: 0.9563
5/5 [==============================] - 0s 2ms/step - loss: 0.1612 - accurac
y: 0.9563
5/5 [==============================] - 0s 2ms/step - loss: 0.1776 - accurac
y: 0.9500
5/5 [==============================] - 0s 2ms/step - loss: 0.1994 - accurac
y: 0.9500
5/5 [==============================] - 0s 2ms/step - loss: 0.1676 - accurac
y: 0.9563
Accuracy: 95.25%
```

In [13]:

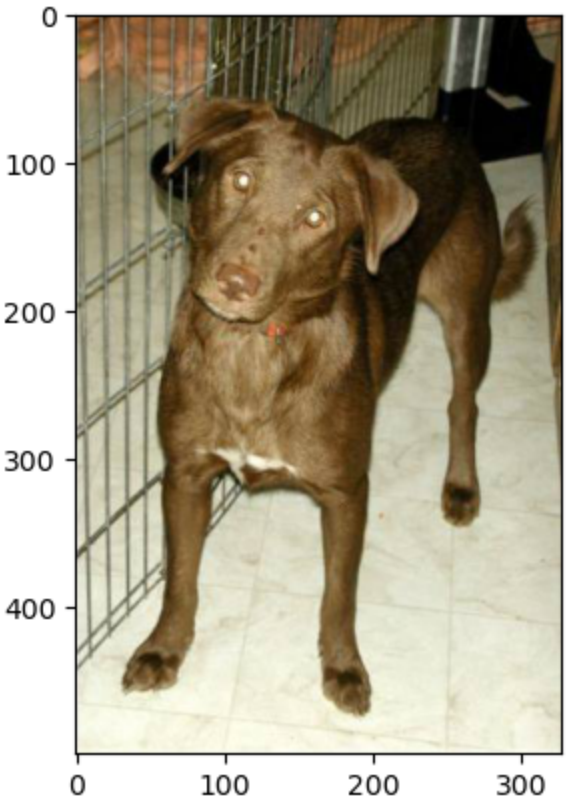
```
###  (4) two-hidden layer NN with 50 hidden nodes (1st layer) and 10 hidden

kfold = StratifiedKFold(n_splits=10, shuffle=True)
accuracy = []


for train_index, test_index in kfold.split(x,y):
    model = keras.Sequential([
    keras.layers.Flatten(input_shape=(64, 64, 1),name="Input"),
    keras.layers.Dense(50, activation='sigmoid',name="Hidden_1"),
    keras.layers.Dense(10, activation='sigmoid',name="Hidden_2"),
    keras.layers.Dense(40, name="Output"),
    ])

    model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits
              metrics=['accuracy'])

    model.fit(x_train, y_train, epochs=300, batch_size=32, verbose = 0)
    scores = model.evaluate(x_test, y_test)
    accuracy.append(scores[1] * 100)


print("Accuracy: %.2f%%" % (np.mean(accuracy)))
```

```
5/5 [==============================] - 0s 1ms/step - loss: 2.1896 - accurac
y: 0.3125
5/5 [==============================] - 0s 1ms/step - loss: 2.8938 - accurac
y: 0.1688
5/5 [==============================] - 0s 1ms/step - loss: 2.0820 - accurac
y: 0.5500
5/5 [==============================] - 0s 1ms/step - loss: 2.7719 - accurac
y: 0.1437
5/5 [==============================] - 0s 2ms/step - loss: 2.2514 - accurac
y: 0.3750
5/5 [==============================] - 0s 2ms/step - loss: 2.6281 - accurac
y: 0.2313
5/5 [==============================] - 0s 2ms/step - loss: 1.7763 - accurac
y: 0.6313
5/5 [==============================] - 0s 2ms/step - loss: 2.5160 - accurac
y: 0.2875
5/5 [==============================] - 0s 2ms/step - loss: 2.7381 - accurac
y: 0.1625
5/5 [==============================] - 0s 6ms/step - loss: 2.6459 - accurac
y: 0.2000
Accuracy: 30.63%
```

Question 7 [3 pts]: Please follow “Dog Cat Dense Layer Classification [Notebook, html] ” to create a dense layer network image classifiers and validate its performances.Please download the cat vs. dog images (dogvscat1000.zip) from Canvas (You can also download whole image dataset from the Kaggle site (https://www.kaggle.com/c/dogs-vs-cats/data). Unzip the downloaded (zip) file. There are 1000 dog and cat images (500 for each category) in the unzipped folder.Use 10-fold cross validation to split the 1000 images into training vs. test set. Train neural networks using training set, and report their performance on the test set.

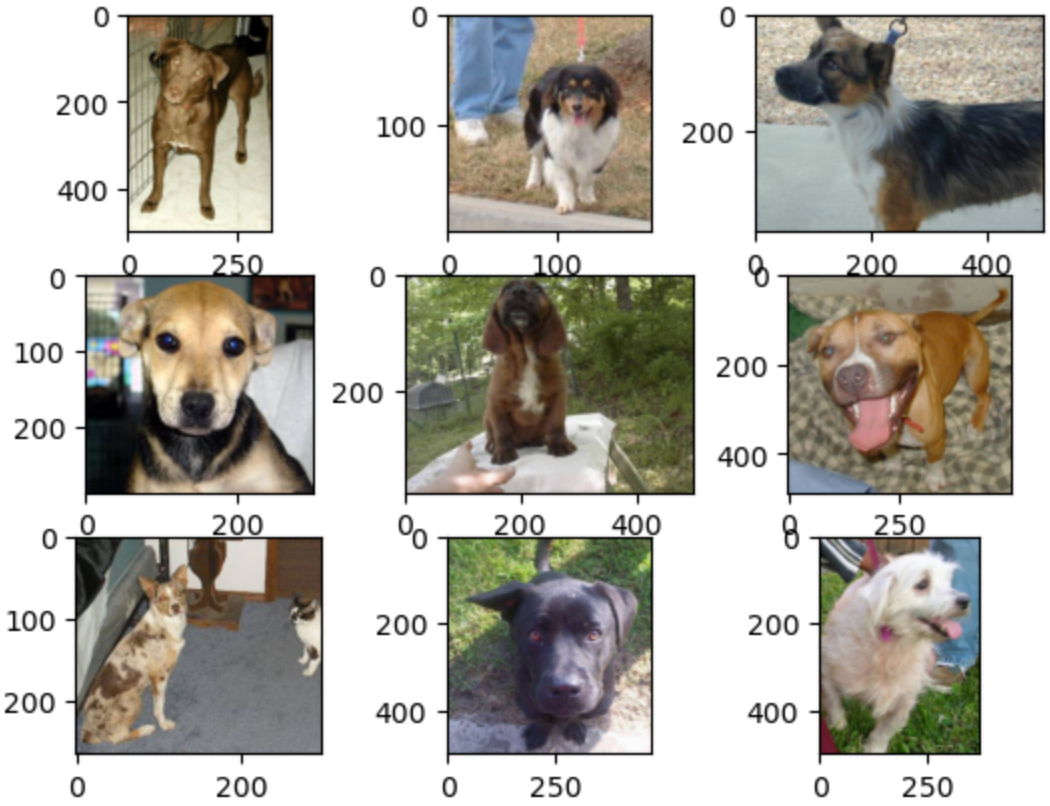
In [14]:
```
folder='dogvscat1000/'
# show one image
filename = folder + 'dog.' + '1' + '.jpg'
# load image pixels
image = imread(filename)
# plot raw pixel data
plt.imshow(image)
```

Out[14]:
```
<matplotlib.image.AxesImage at 0x7a9fa5197c40>
```

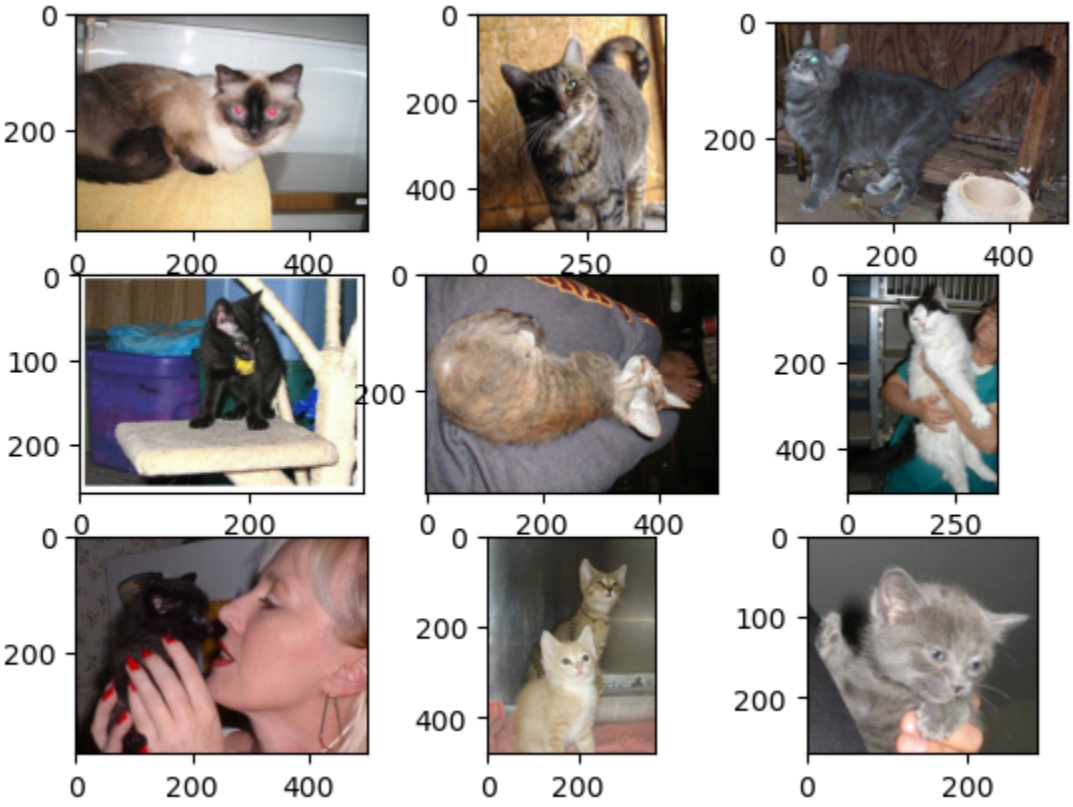


In [15]:
```
def displayImages(foldername,dogorcat,startID):
    # plot first few images
    for i in range(9):
        #define subplot 3x3
        plt.subplot(330 + 1 + i)
        # define filename
        filename = foldername + dogorcat +'.' + str(i+startID) + '.jpg'
        # load image pixels
        image = imread(filename)
        # plot raw pixel data
        plt.imshow(image)
        # show the figure
plt.show()
```

In [16]:
```
displayImages(folder,"dog",1)
```



In [17]:
```
displayImages(folder,"cat",20)
```



In [18]:
```
folder = 'dogvscat1000/'
photos, labels = list(), list()
# enumerate files in the directory
```

```
for file in listdir(folder):
    # determine class
    output = 0.0
    if file.startswith('cat'):
        output = 1.0
        # load image
    photo = load_img(folder + file, target_size=(32, 32),color_mode="rgb")
    # convert to numpy array
    photo = img_to_array(photo)
    # store
    photos.append(photo)
    labels.append(output)
# convert to a numpy arrays
photos = asarray(photos)
labels = asarray(labels)
print(photos.shape, labels.shape)
# save the reshaped photos
save('dogs_vs_cats_photos.npy', photos)
save('dogs_vs_cats_labels.npy', labels)
```

```
(1000, 32, 32, 3) (1000,)
```

In [19]:
```
print(photos[1].shape)
```

```
(32, 32, 3)
```

In [20]:
```
X_train, X_test, y_train, y_test = train_test_split(photos, labels, test_siz
```

Create a feedforward neural network with one or two hidden layers (you can determine the number of hidden nodes for each layer, but the number of hidden nodes for each layer should be at least 50). Train the network on the training set and report its performance on the test. Report at least one loss plot with respect to the number of epochs. Report 10-fold cross validation accuracy, including mean and standard deviation [1 pt]

In [21]:
```
model = keras.Sequential()
model.add(layers.LayerNormalization(axis=3 , center=True , scale=True))
model.add(layers.Flatten(input_shape=(32, 32, 3),name="Input"))
model.add(layers.Dense(256, activation='relu',name="Hidden"))
model.add(layers.Dense(100, activation='relu',name="Hidden2"))
model.add(layers.Dense(50, activation='relu',name="Hidden3"))
model.add(layers.Dense(2, name="Output"))
```

In [22]:
```
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits
              metrics=['accuracy'])
```

In [23]:
```
history=model.fit(X_train, y_train, epochs=100,verbose=1,batch_size=32)
```

```
Epoch 1/100
19/19 [==============================] - 1s 4ms/step - loss: 0.7639 - accura
cy: 0.5983
Epoch 2/100
19/19 [==============================] - 0s 4ms/step - loss: 0.4820 - accura
cy: 0.7817
Epoch 3/100
19/19 [==============================] - 0s 4ms/step - loss: 0.3484 - accura
cy: 0.8417
Epoch 4/100
19/19 [==============================] - 0s 6ms/step - loss: 0.2278 - accura
cy: 0.9033
Epoch 5/100
19/19 [==============================] - 0s 10ms/step - loss: 0.2064 - accur
acy: 0.9050
Epoch 6/100
19/19 [==============================] - 0s 12ms/step - loss: 0.1706 - accur
acy: 0.9283
Epoch 7/100
19/19 [==============================] - 0s 12ms/step - loss: 0.1795 - accur
acy: 0.9217
Epoch 8/100
19/19 [==============================] - 0s 13ms/step - loss: 0.1833 - accur
acy: 0.9200
Epoch 9/100
19/19 [==============================] - 0s 13ms/step - loss: 0.0730 - accur
acy: 0.9733
Epoch 10/100
19/19 [==============================] - 0s 12ms/step - loss: 0.0644 - accur
acy: 0.9750
Epoch 11/100
19/19 [==============================] - 0s 12ms/step - loss: 0.0328 - accur
acy: 0.9950
Epoch 12/100
19/19 [==============================] - 0s 12ms/step - loss: 0.0287 - accur
acy: 0.9933
Epoch 13/100
19/19 [==============================] - 0s 14ms/step - loss: 0.0862 - accur
acy: 0.9700
Epoch 14/100
19/19 [==============================] - 0s 13ms/step - loss: 0.1465 - accur
acy: 0.9367
Epoch 15/100
19/19 [==============================] - 0s 11ms/step - loss: 0.1030 - accur
acy: 0.9600
Epoch 16/100
19/19 [==============================] - 0s 13ms/step - loss: 0.0718 - accur
acy: 0.9800
Epoch 17/100
19/19 [==============================] - 0s 13ms/step - loss: 0.1052 - accur
acy: 0.9650
Epoch 18/100
19/19 [==============================] - 0s 12ms/step - loss: 0.1193 - accur
acy: 0.9600
Epoch 19/100
19/19 [==============================] - 0s 12ms/step - loss: 0.1328 - accur
```

```
acy: 0.9683
Epoch 20/100
19/19 [==============================] - 0s 12ms/step - loss: 0.0656 - accur
acy: 0.9833
Epoch 21/100
19/19 [==============================] - 0s 12ms/step - loss: 0.0408 - accur
acy: 0.9867
Epoch 22/100
19/19 [==============================] - 0s 12ms/step - loss: 0.0096 - accur
acy: 0.9983
Epoch 23/100
19/19 [==============================] - 0s 12ms/step - loss: 0.0043 - accur
acy: 1.0000
Epoch 24/100
19/19 [==============================] - 0s 13ms/step - loss: 0.0047 - accur
acy: 1.0000
Epoch 25/100
19/19 [==============================] - 0s 13ms/step - loss: 0.0056 - accur
acy: 1.0000
Epoch 26/100
19/19 [==============================] - 0s 14ms/step - loss: 0.0026 - accur
acy: 1.0000
Epoch 27/100
19/19 [==============================] - 0s 13ms/step - loss: 0.0016 - accur
acy: 1.0000
Epoch 28/100
19/19 [==============================] - 0s 12ms/step - loss: 0.0019 - accur
acy: 1.0000
Epoch 29/100
19/19 [==============================] - 0s 12ms/step - loss: 0.0012 - accur
acy: 1.0000
Epoch 30/100
19/19 [==============================] - 0s 12ms/step - loss: 9.4767e-04 - a
ccuracy: 1.0000
Epoch 31/100
19/19 [==============================] - 0s 12ms/step - loss: 8.2189e-04 - a
ccuracy: 1.0000
Epoch 32/100
19/19 [==============================] - 0s 11ms/step - loss: 8.2736e-04 - a
ccuracy: 1.0000
Epoch 33/100
19/19 [==============================] - 0s 12ms/step - loss: 6.7393e-04 - a
ccuracy: 1.0000
Epoch 34/100
19/19 [==============================] - 0s 13ms/step - loss: 7.1028e-04 - a
ccuracy: 1.0000
Epoch 35/100
19/19 [==============================] - 0s 12ms/step - loss: 5.6504e-04 - a
ccuracy: 1.0000
Epoch 36/100
19/19 [==============================] - 0s 12ms/step - loss: 6.3693e-04 - a
ccuracy: 1.0000
Epoch 37/100
19/19 [==============================] - 0s 12ms/step - loss: 6.2161e-04 - a
ccuracy: 1.0000
Epoch 38/100
```

```
19/19 [==============================] - 0s 13ms/step - loss: 4.9056e-04 - a
ccuracy: 1.0000
Epoch 39/100
19/19 [==============================] - 0s 12ms/step - loss: 4.5828e-04 - a
ccuracy: 1.0000
Epoch 40/100
19/19 [==============================] - 0s 12ms/step - loss: 5.1233e-04 - a
ccuracy: 1.0000
Epoch 41/100
19/19 [==============================] - 0s 12ms/step - loss: 5.0272e-04 - a
ccuracy: 1.0000
Epoch 42/100
19/19 [==============================] - 0s 15ms/step - loss: 3.8555e-04 - a
ccuracy: 1.0000
Epoch 43/100
19/19 [==============================] - 0s 13ms/step - loss: 5.4342e-04 - a
ccuracy: 1.0000
Epoch 44/100
19/19 [==============================] - 0s 12ms/step - loss: 3.7162e-04 - a
ccuracy: 1.0000
Epoch 45/100
19/19 [==============================] - 0s 11ms/step - loss: 3.3817e-04 - a
ccuracy: 1.0000
Epoch 46/100
19/19 [==============================] - 0s 12ms/step - loss: 3.3681e-04 - a
ccuracy: 1.0000
Epoch 47/100
19/19 [==============================] - 0s 12ms/step - loss: 3.2704e-04 - a
ccuracy: 1.0000
Epoch 48/100
19/19 [==============================] - 0s 11ms/step - loss: 3.0336e-04 - a
ccuracy: 1.0000
Epoch 49/100
19/19 [==============================] - 0s 11ms/step - loss: 2.8747e-04 - a
ccuracy: 1.0000
Epoch 50/100
19/19 [==============================] - 0s 12ms/step - loss: 2.7408e-04 - a
ccuracy: 1.0000
Epoch 51/100
19/19 [==============================] - 0s 12ms/step - loss: 2.5256e-04 - a
ccuracy: 1.0000
Epoch 52/100
19/19 [==============================] - 0s 12ms/step - loss: 2.9144e-04 - a
ccuracy: 1.0000
Epoch 53/100
19/19 [==============================] - 0s 12ms/step - loss: 2.4782e-04 - a
ccuracy: 1.0000
Epoch 54/100
19/19 [==============================] - 0s 12ms/step - loss: 2.4996e-04 - a
ccuracy: 1.0000
Epoch 55/100
19/19 [==============================] - 0s 11ms/step - loss: 3.4247e-04 - a
ccuracy: 1.0000
Epoch 56/100
19/19 [==============================] - 0s 13ms/step - loss: 2.1454e-04 - a
ccuracy: 1.0000
```

```
Epoch 57/100
19/19 [==============================] - 0s 12ms/step - loss: 2.0454e-04 - a
ccuracy: 1.0000
Epoch 58/100
19/19 [==============================] - 0s 11ms/step - loss: 2.0979e-04 - a
ccuracy: 1.0000
Epoch 59/100
19/19 [==============================] - 0s 13ms/step - loss: 1.9719e-04 - a
ccuracy: 1.0000
Epoch 60/100
19/19 [==============================] - 0s 12ms/step - loss: 2.0018e-04 - a
ccuracy: 1.0000
Epoch 61/100
19/19 [==============================] - 0s 12ms/step - loss: 2.0280e-04 - a
ccuracy: 1.0000
Epoch 62/100
19/19 [==============================] - 0s 13ms/step - loss: 1.6652e-04 - a
ccuracy: 1.0000
Epoch 63/100
19/19 [==============================] - 0s 12ms/step - loss: 1.6364e-04 - a
ccuracy: 1.0000
Epoch 64/100
19/19 [==============================] - 0s 13ms/step - loss: 1.5462e-04 - a
ccuracy: 1.0000
Epoch 65/100
19/19 [==============================] - 0s 13ms/step - loss: 1.5520e-04 - a
ccuracy: 1.0000
Epoch 66/100
19/19 [==============================] - 0s 12ms/step - loss: 1.5505e-04 - a
ccuracy: 1.0000
Epoch 67/100
19/19 [==============================] - 0s 13ms/step - loss: 1.5938e-04 - a
ccuracy: 1.0000
Epoch 68/100
19/19 [==============================] - 0s 13ms/step - loss: 1.6871e-04 - a
ccuracy: 1.0000
Epoch 69/100
19/19 [==============================] - 0s 12ms/step - loss: 1.3536e-04 - a
ccuracy: 1.0000
Epoch 70/100
19/19 [==============================] - 0s 12ms/step - loss: 1.2692e-04 - a
ccuracy: 1.0000
Epoch 71/100
19/19 [==============================] - 0s 12ms/step - loss: 1.2840e-04 - a
ccuracy: 1.0000
Epoch 72/100
19/19 [==============================] - 0s 12ms/step - loss: 1.2658e-04 - a
ccuracy: 1.0000
Epoch 73/100
19/19 [==============================] - 0s 12ms/step - loss: 1.2358e-04 - a
ccuracy: 1.0000
Epoch 74/100
19/19 [==============================] - 0s 13ms/step - loss: 1.3511e-04 - a
ccuracy: 1.0000
Epoch 75/100
19/19 [==============================] - 0s 12ms/step - loss: 1.1402e-04 - a
```

```
ccuracy: 1.0000
Epoch 76/100
19/19 [==============================] - 0s 11ms/step - loss: 1.1014e-04 - a
ccuracy: 1.0000
Epoch 77/100
19/19 [==============================] - 0s 12ms/step - loss: 1.0751e-04 - a
ccuracy: 1.0000
Epoch 78/100
19/19 [==============================] - 0s 12ms/step - loss: 1.0323e-04 - a
ccuracy: 1.0000
Epoch 79/100
19/19 [==============================] - 0s 12ms/step - loss: 1.0079e-04 - a
ccuracy: 1.0000
Epoch 80/100
19/19 [==============================] - 0s 12ms/step - loss: 1.0476e-04 - a
ccuracy: 1.0000
Epoch 81/100
19/19 [==============================] - 0s 12ms/step - loss: 9.4577e-05 - a
ccuracy: 1.0000
Epoch 82/100
19/19 [==============================] - 0s 12ms/step - loss: 9.4378e-05 - a
ccuracy: 1.0000
Epoch 83/100
19/19 [==============================] - 0s 12ms/step - loss: 8.8374e-05 - a
ccuracy: 1.0000
Epoch 84/100
19/19 [==============================] - 0s 12ms/step - loss: 8.7274e-05 - a
ccuracy: 1.0000
Epoch 85/100
19/19 [==============================] - 0s 12ms/step - loss: 8.4166e-05 - a
ccuracy: 1.0000
Epoch 86/100
19/19 [==============================] - 0s 12ms/step - loss: 8.1765e-05 - a
ccuracy: 1.0000
Epoch 87/100
19/19 [==============================] - 0s 12ms/step - loss: 8.2604e-05 - a
ccuracy: 1.0000
Epoch 88/100
19/19 [==============================] - 0s 12ms/step - loss: 7.5869e-05 - a
ccuracy: 1.0000
Epoch 89/100
19/19 [==============================] - 0s 12ms/step - loss: 7.5382e-05 - a
ccuracy: 1.0000
Epoch 90/100
19/19 [==============================] - 0s 12ms/step - loss: 7.1761e-05 - a
ccuracy: 1.0000
Epoch 91/100
19/19 [==============================] - 0s 12ms/step - loss: 7.2399e-05 - a
ccuracy: 1.0000
Epoch 92/100
19/19 [==============================] - 0s 12ms/step - loss: 7.3811e-05 - a
ccuracy: 1.0000
Epoch 93/100
19/19 [==============================] - 0s 13ms/step - loss: 7.3064e-05 - a
ccuracy: 1.0000
Epoch 94/100
```

```
19/19 [==============================] - 0s 13ms/step - loss: 7.1036e-05 - a
ccuracy: 1.0000
Epoch 95/100
19/19 [==============================] - 0s 12ms/step - loss: 6.5318e-05 - a
ccuracy: 1.0000
Epoch 96/100
19/19 [==============================] - 0s 12ms/step - loss: 6.3086e-05 - a
ccuracy: 1.0000
Epoch 97/100
19/19 [==============================] - 0s 11ms/step - loss: 6.4075e-05 - a
ccuracy: 1.0000
Epoch 98/100
19/19 [==============================] - 0s 13ms/step - loss: 6.1482e-05 - a
ccuracy: 1.0000
Epoch 99/100
19/19 [==============================] - 0s 12ms/step - loss: 5.8507e-05 - a
ccuracy: 1.0000
Epoch 100/100
19/19 [==============================] - 0s 12ms/step - loss: 5.7097e-05 - a
ccuracy: 1.0000
```

In [24]:
```
model.summary()
```

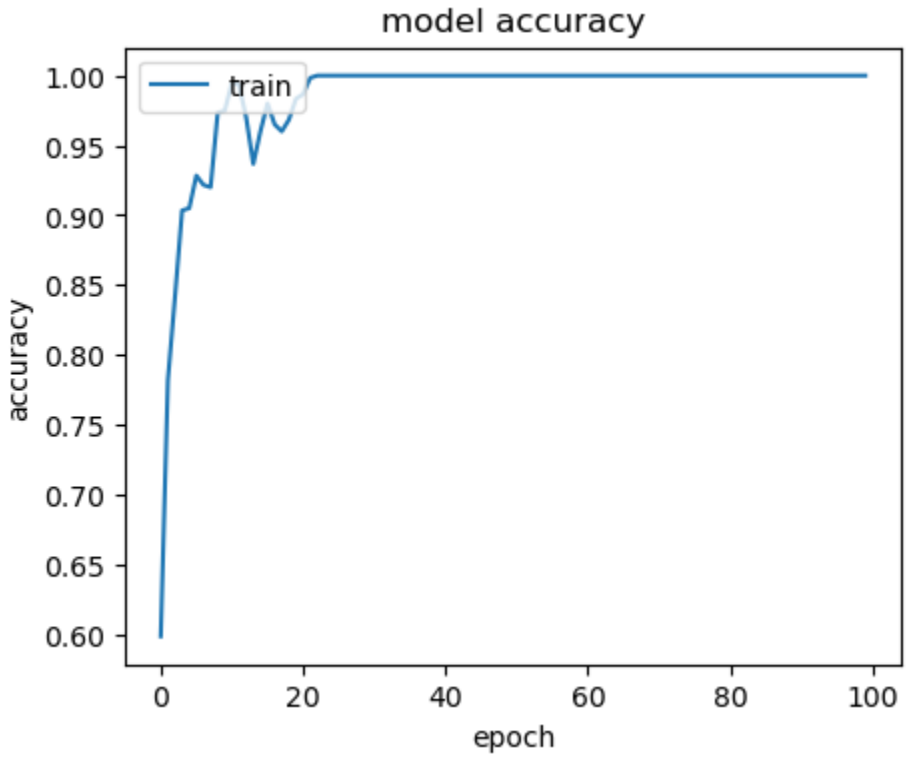
```
Model: "sequential_41"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 layer_normalization (LayerN  (None, 32, 32, 3)        6
 ormalization)

 Input (Flatten)             (None, 3072)              0

 Hidden (Dense)              (None, 256)               786688

 Hidden2 (Dense)             (None, 100)               25700

 Hidden3 (Dense)             (None, 50)                5050

 Output (Dense)              (None, 2)                 102

=================================================================
Total params: 817,546
Trainable params: 817,546
Non-trainable params: 0
_________________________________________________________________
```

In [25]:
```
test_loss, test_acc = model.evaluate(X_test,  y_test, verbose=2)

print('\nTest accuracy:', test_acc)
```

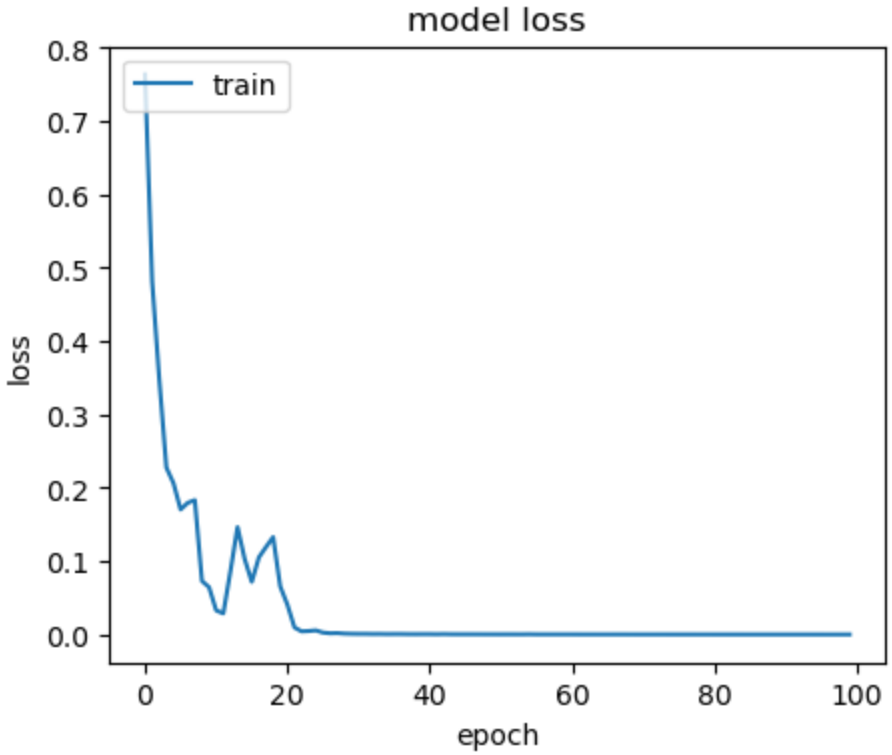
```
13/13 - 0s - loss: 2.2089 - accuracy: 0.6125 - 206ms/epoch - 16ms/step

Test accuracy: 0.612500011920929
```

In [26]:
```
plt.figure(figsize = [5,4])

plt.plot(history.history['accuracy'])
```

```
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'validation'], loc='upper left')
plt.show()
```



In [27]:
```
plt.figure(figsize = [5,4])

plt.plot(history.history['loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'validation'], loc='upper left')
plt.show()
```



In [28]:
```
kfold = StratifiedKFold(n_splits=10, shuffle=True)
accuracy = []


for train_index, test_index in kfold.split(x,y):
    model = keras.Sequential()
    model.add(layers.LayerNormalization(axis=3 , center=True , scale=True))
    model.add(layers.Flatten(input_shape=(32, 32, 3),name="Input"))
    model.add(layers.Dense(256, activation='relu',name="Hidden"))
    model.add(layers.Dense(100, activation='relu',name="Hidden2"))
    model.add(layers.Dense(50, activation='relu',name="Hidden3"))
    model.add(layers.Dense(2, name="Output"))
    model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits
              metrics=['accuracy'])

    model.fit(X_train, y_train, epochs=300, batch_size=32, verbose = 0)
    scores = model.evaluate(X_test, y_test)
    accuracy.append(scores[1] * 100)


print("Accuracy: %.2f%% (+/- %.2f%%)" % (np.mean(accuracy), np.std(accuracy)
```

```
13/13 [==============================] - 0s 2ms/step - loss: 3.7884 - accura
cy: 0.6225
13/13 [==============================] - 0s 3ms/step - loss: 2.8989 - accura
cy: 0.6000
13/13 [==============================] - 0s 2ms/step - loss: 2.7634 - accura
cy: 0.5700
13/13 [==============================] - 0s 3ms/step - loss: 3.1992 - accura
cy: 0.6000
13/13 [==============================] - 0s 3ms/step - loss: 2.6217 - accura
cy: 0.6050
13/13 [==============================] - 0s 3ms/step - loss: 2.7260 - accura
cy: 0.6175
13/13 [==============================] - 0s 2ms/step - loss: 2.8840 - accura
cy: 0.6075
13/13 [==============================] - 0s 3ms/step - loss: 2.8335 - accura
cy: 0.6250
13/13 [==============================] - 0s 4ms/step - loss: 2.9669 - accura
cy: 0.5925
13/13 [==============================] - 0s 3ms/step - loss: 3.0194 - accura
cy: 0.5950
Accuracy: 60.35% (+/- 1.54%)
```

For the same network structure created above, please change the network structure by adding batch normalization and/or dropout layer (at different layers). Report 10-fold cross validation performance of the network by adding batch normalization and dropout layer, respectively (you can decide which layer(s) to add batch normalization and/or dropout). Conclude batch normalization and drop-out impact on the neural network performance. [2 pts]

> Batch normalization and Dropout are added to the neural network so that overfitting doesnt occur. Overfitting occurs when the model becomes too complex for the data and while it performs very well with known data, it fails to do so for unknown data. For this neural network, there's hardly any change in accuracy due to addition of batch normalization and Dropout layer since the model isn't complex enough to overfit.

In [29]:
```
kfold = StratifiedKFold(n_splits=10, shuffle=True)
accuracy = []


for train_index, test_index in kfold.split(x,y):
    model = keras.Sequential()
    model.add(layers.LayerNormalization(axis=3 , center=True , scale=True))
    model.add(layers.Flatten(input_shape=(32, 32, 3),name="Input"))

    model.add(layers.Dense(256, activation='relu',name="Hidden"))

    model.add(layers.Dense(100, activation='relu',name="Hidden2"))
    model.add(keras.layers.BatchNormalization()),

    model.add(layers.Dense(50, activation='relu',name="Hidden3"))
```

```
    model.add(keras.layers.Dropout(rate=0.3)),

    model.add(layers.Dense(2, name="Output"))
    model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits
              metrics=['accuracy'])

    model.fit(X_train, y_train, epochs=300, batch_size=32, verbose = 0)
    scores = model.evaluate(X_test, y_test)
    accuracy.append(scores[1] * 100)


print("Accuracy: %.2f%% (+/- %.2f%%)" % (np.mean(accuracy), np.std(accuracy)
```

```
13/13 [==============================] - 0s 3ms/step - loss: 2.7182 - accura
cy: 0.5800
13/13 [==============================] - 0s 3ms/step - loss: 2.5336 - accura
cy: 0.5650
13/13 [==============================] - 0s 2ms/step - loss: 3.4185 - accura
cy: 0.6125
13/13 [==============================] - 0s 3ms/step - loss: 3.0619 - accura
cy: 0.6000
13/13 [==============================] - 0s 2ms/step - loss: 2.3669 - accura
cy: 0.6025
13/13 [==============================] - 0s 2ms/step - loss: 2.3677 - accura
cy: 0.6200
13/13 [==============================] - 0s 2ms/step - loss: 2.6489 - accura
cy: 0.5925
13/13 [==============================] - 0s 2ms/step - loss: 3.4103 - accura
cy: 0.6025
13/13 [==============================] - 0s 3ms/step - loss: 2.4280 - accura
cy: 0.5950
13/13 [==============================] - 0s 2ms/step - loss: 2.7606 - accura
cy: 0.6025
Accuracy: 59.73% (+/- 1.49%)
```

In [ ]:

In [ ]:

In [ ]: