HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY AND EDUCATION

FACULTY FOR HIGH QUALITY TRAINING

MAJOR: ARTIFICIAL INTELLIGENCE

ANN TRAINING HOMEWORK

NAME: **TRAN QUOC TOAN**

STUDENT ID: **19146090**

ADVISOR: **NGUYEN TRUONG THINH, Ph.D**

Ho Chi Minh City, 18th of May 2022

TABLE OF CONTENTS

EXERCISE 1: CIFAR10

```
from keras.datasets import cifar10
```

In [ ]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
```

In [ ]:

```
from tensorflow.keras.utils import to_categorical
from keras.models import Sequential
from keras.layers import Dense,Activation,Dropout
from tensorflow.keras.optimizers import RMSprop
from keras.callbacks import EarlyStopping
```

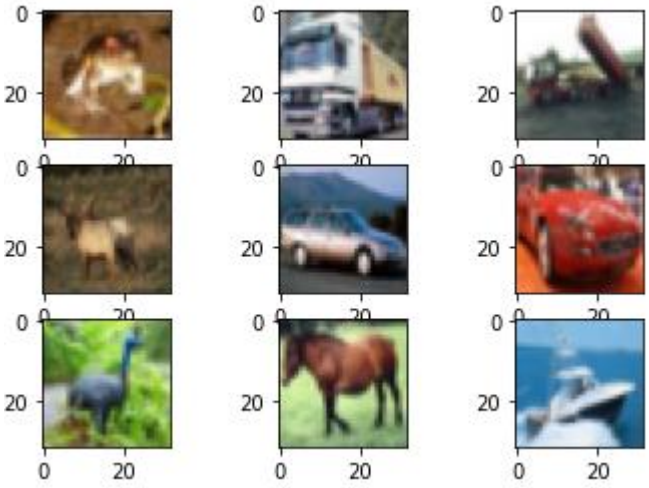
In [ ]:

```
(x_train, y_train), (x_test, y_test) = cifar10.load_data()
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170500096/170498071 [==============================] - 2s 0us/step
170508288/170498071 [==============================] - 2s 0us/step
```

In [ ]:

```
for i in range(9):
  plt.subplot(330+i+1)
  plt.imshow(x_train[i], cmap = plt.get_cmap('gray'))
plt.show()
```



In [ ]:

```
x_train = x_train.reshape(x_train.shape[0],-1)
x_test = x_test.reshape(x_test.shape[0],-1)
```

In [ ]:

```
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
```

In [ ]:

```
x_train /= 255
```

```
x_test /= 255
```

In [ ]:

```
x_train.shape
```

Out[ ]:

```
(50000, 3072)
```

In [ ]:

```
y_train = to_categorical(y_train,10)
y_test = to_categorical(y_test,10)
```

In [ ]:

```
model = Sequential()
model.add(Dense(512,activation= 'relu' ,input_shape=(3072,)))
model.add(Dense(512,activation='relu'))
model.add(Dense(10,activation='softmax'))
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 512)               1573376

 dense_1 (Dense)             (None, 512)               262656

 dense_2 (Dense)             (None, 10)                5130

=================================================================
Total params: 1,841,162
Trainable params: 1,841,162
Non-trainable params: 0
_________________________________________________________________
```

In [ ]:

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

In [ ]:

```
history = model.fit(x_train, y_train, batch_size = 128, epochs = 200, verbose
=1, validation_data = (x_test, y_test))
```

```
Epoch 1/200
391/391 [==============================] - 17s 42ms/step - loss: 2.1962 - acc
uracy: 0.2567 - val_loss: 1.8665 - val_accuracy: 0.3352
Epoch 2/200
391/391 [==============================] - 18s 47ms/step - loss: 1.7775 - acc
uracy: 0.3607 - val_loss: 1.7801 - val_accuracy: 0.3515

...

Epoch 197/200
391/391 [==============================] - 16s 41ms/step - loss: 0.7590 - acc
uracy: 0.7553 - val_loss: 4.4574 - val_accuracy: 0.4503
Epoch 198/200
391/391 [==============================] - 16s 41ms/step - loss: 0.7696 - acc
uracy: 0.7547 - val_loss: 4.6529 - val_accuracy: 0.4403
Epoch 199/200
```

```
391/391 [==============================] - 16s 41ms/step - loss: 0.7736 - acc
uracy: 0.7513 - val_loss: 4.3999 - val_accuracy: 0.4443
Epoch 200/200
391/391 [==============================] - 16s 41ms/step - loss: 0.7838 - acc
uracy: 0.7518 - val_loss: 4.2500 - val_accuracy: 0.4677
```

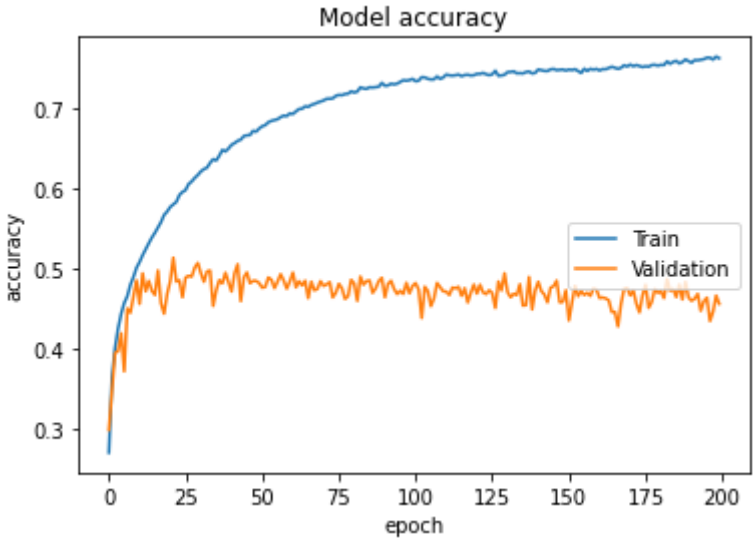
In [ ]:

```
Score = model.evaluate(x_test,y_test,verbose = 1)
```

```
313/313 [==============================] - 1s 2ms/step - loss: 4.5000 - accur
acy: 0.4563
```

In [ ]:

```
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 ='center right')
plt.show()
```



EXERCISE 2: CIFAR100

```
from keras.datasets import cifar100
```

In [ ]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
```

In [ ]:

```
from tensorflow.keras.utils import to_categorical
from keras.models import Sequential
from keras.layers import Dense,Activation,Dropout
from tensorflow.keras.optimizers import RMSprop
from keras.callbacks import EarlyStopping
```

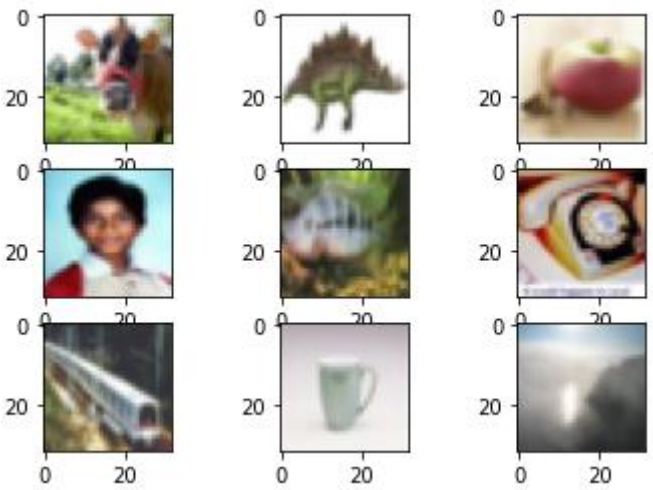
In [ ]:

```
(x_train, y_train), (x_test, y_test) = cifar100.load_data()
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-100-python.tar.g
z
169009152/169001437 [==============================] - 2s 0us/step
169017344/169001437 [==============================] - 2s 0us/step
```

In [ ]:

```
for i in range(9):
  plt.subplot(330+i+1)
  plt.imshow(x_train[i], cmap = plt.get_cmap('gray'))
plt.show()
```



```
x_train = x_train.reshape(x_train.shape[0],-1)
x_test = x_test.reshape(x_test.shape[0],-1)
```

In [ ]:

```
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
```

In [ ]:

```
x_train /= 255
x_test /= 255
```

In [ ]:

```
x_train.shape
```

Out[ ]:

```
(50000, 3072)
```

In [ ]:

```
y_train = to_categorical(y_train,100)
y_test = to_categorical(y_test,100)
```

In [ ]:

```
model = Sequential()
model.add(Dense(512,activation= 'relu' ,input_shape=(3072,)))
model.add(Dense(512,activation='relu'))
model.add(Dense(100,activation='softmax'))
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 512)               1573376

 dense_1 (Dense)             (None, 512)               262656

 dense_2 (Dense)             (None, 100)               51300

=================================================================
Total params: 1,887,332
Trainable params: 1,887,332
Non-trainable params: 0
_________________________________________________________________
```

In [ ]:

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

In [ ]:

```
history = model.fit(x_train, y_train, batch_size = 128, epochs = 200, verbose
=1, validation_data = (x_test, y_test))
```

```
Epoch 1/200
391/391 [==============================] - 5s 6ms/step - loss: 4.2159 - accur
acy: 0.0676 - val_loss: 4.0825 - val_accuracy: 0.0799
Epoch 2/200
391/391 [==============================] - 2s 4ms/step - loss: 3.7439 - accur
acy: 0.1300 - val_loss: 3.7979 - val_accuracy: 0.1307

…

Epoch 198/200
391/391 [==============================] - 2s 5ms/step - loss: 3.6271 - accur
acy: 0.1711 - val_loss: 4.2361 - val_accuracy: 0.1252
Epoch 199/200
391/391 [==============================] - 2s 5ms/step - loss: 3.6606 - accur
acy: 0.1708 - val_loss: 4.1416 - val_accuracy: 0.1402
```

```
Epoch 200/200
391/391 [==============================] - 2s 4ms/step - loss: 3.6024 - accur
acy: 0.1714 - val_loss: 4.1918 - val_accuracy: 0.1335
```

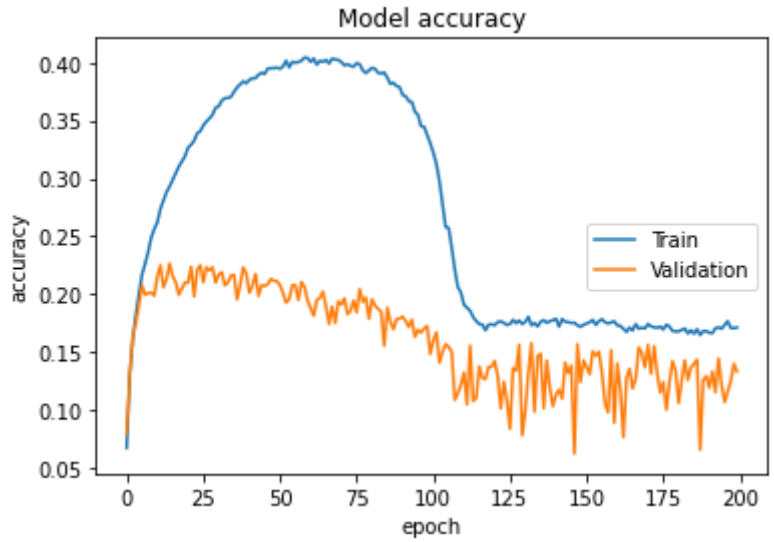
In [ ]:

```
Score = model.evaluate(x_test,y_test,verbose = 1)
```

```
313/313 [==============================] - 1s 3ms/step - loss: 4.1918 - accur
acy: 0.1335
```

In [ ]:

```
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 ='center right')
plt.show()
```



EXERCISE 3: FASHION_MNIST

```
from keras.datasets import fashion_mnist
```

In [ ]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
```

In [ ]:

```
from tensorflow.keras.utils import to_categorical
from keras.models import Sequential
from keras.layers import Dense,Activation,Dropout
from tensorflow.keras.optimizers import RMSprop
from keras.callbacks import EarlyStopping
```

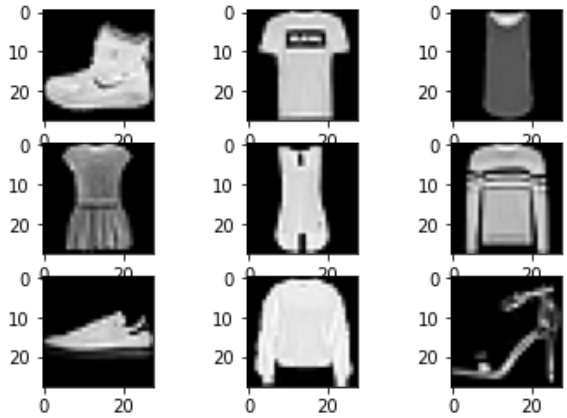
In [ ]:

```
(x_train, y_train), (x_test, y_test) = fashion_mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-data
sets/train-labels-idx1-ubyte.gz
32768/29515 [================================] - 0s 0us/step
40960/29515 [========================================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-data
sets/train-images-idx3-ubyte.gz
26427392/26421880 [==============================] - 0s 0us/step
26435584/26421880 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-data
sets/t10k-labels-idx1-ubyte.gz
16384/5148 [===================================================================
==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-data
sets/t10k-images-idx3-ubyte.gz
4423680/4422102 [==============================] - 0s 0us/step
4431872/4422102 [==============================] - 0s 0us/step
```

In [ ]:

```
for i in range(9):
  plt.subplot(330+i+1)
  plt.imshow(x_train[i], cmap = plt.get_cmap('gray'))
plt.show()
```



```
x_train.shape
```

Out[ ]:

```
(60000, 28, 28)
```

In [ ]:

```
x_train = x_train.reshape(x_train.shape[0],-1)
x_test = x_test.reshape(x_test.shape[0],-1)
```

In [ ]:

```
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
```

In [ ]:

```
x_train /= 255
x_test /= 255
```

In [ ]:

```
y_train = to_categorical(y_train,10)
y_test = to_categorical(y_test,10)
```

In [ ]:

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

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 dense (Dense)                (None, 512)               401920

 dense_1 (Dense)              (None, 512)               262656

 dense_2 (Dense)              (None, 10)                5130

=================================================================
Total params: 669,706
Trainable params: 669,706
Non-trainable params: 0
_________________________________________________________________
```

In [ ]:

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

In [ ]:

```
history = model.fit(x_train, y_train, batch_size = 128, epochs = 200, verbose
=1, validation_data = (x_test, y_test))
```

```
Epoch 1/200
469/469 [==============================] - 5s 4ms/step - loss: 0.5397 - accur
acy: 0.8027 - val_loss: 0.4626 - val_accuracy: 0.8330
Epoch 2/200
469/469 [==============================] - 2s 4ms/step - loss: 0.3744 - accur
acy: 0.8627 - val_loss: 0.4402 - val_accuracy: 0.8399

…

Epoch 197/200
469/469 [==============================] - 2s 4ms/step - loss: 0.0538 - accur
acy: 0.9849 - val_loss: 2.9288 - val_accuracy: 0.8894
Epoch 198/200
469/469 [==============================] - 2s 4ms/step - loss: 0.0677 - accur
acy: 0.9847 - val_loss: 2.8384 - val_accuracy: 0.8910
Epoch 199/200
469/469 [==============================] - 2s 4ms/step - loss: 0.0613 - accur
acy: 0.9845 - val_loss: 3.1128 - val_accuracy: 0.8943
Epoch 200/200
469/469 [==============================] - 2s 4ms/step - loss: 0.0539 - accur
acy: 0.9853 - val_loss: 3.0674 - val_accuracy: 0.8903
```

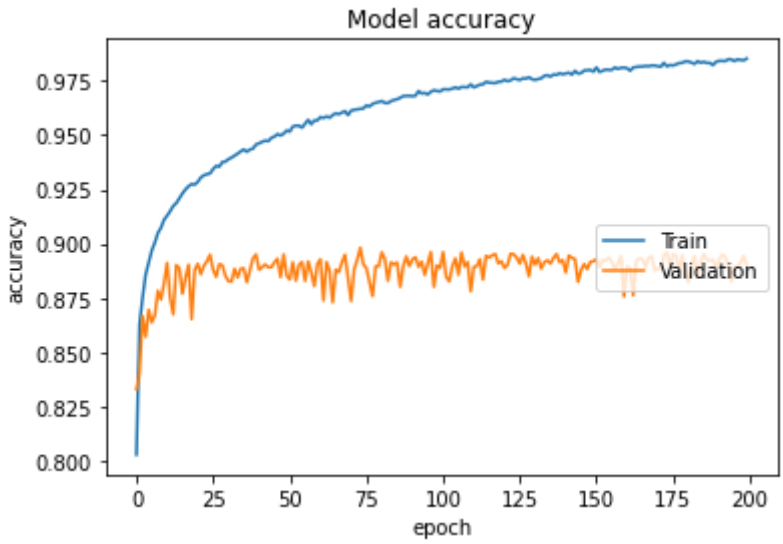
In [ ]:

```
Score = model.evaluate(x_test,y_test,verbose = 1)
```

```
313/313 [==============================] - 1s 2ms/step - loss: 3.0674 - accur
acy: 0.8903
```

In [ ]:

```
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 ='center right')
plt.show()
```


Model accuracy
accuracy
epoch
Train
Validation
0.975
0.950
0.925
0.900
0.875
0.850
0.825
0.800
0
25
50
75
100
125
150
175
200

EXERCISE 4: 2ND DEGREES ROBOTIC ARM

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
```

In [31]:

```
from tensorflow.keras.utils import to_categorical
from keras.models import Sequential
from keras.layers import Dense,Activation,Dropout
from tensorflow.keras.optimizers import RMSprop
from keras.callbacks import EarlyStopping
from sklearn.model_selection import train_test_split
from tensorflow import keras
```

In [32]:

```
l1 = 10
l2 = 40
px = []
py= []
tt1 =[]
tt2 =[]
```

In [33]:

```
for i in np.arange(0, 360, 1):
    for j in np.arange(0, 360, 1):
      x = l1*np.cos(np.radians(i)) + l2*np.cos(np.radians(i+j))
      y = l1*np.sin(np.radians(i)) + l2*np.sin(np.radians(i+j))
      px.append(x)
      py.append(y)
      tt1.append(i)
      tt2.append(j)
```

In [34]:

```
Px = np.array([px])
Py = np.array([py])
Tt1= np.array([tt1])
Tt2 = np.array([tt2])
```

In [35]:

```
Tt1 = Tt1.astype('float32')
Tt2 = Tt2.astype('float32')
```

In [36]:

```
Tt1 /= 360
Tt2 /= 360
```

In [37]:

```
result = np.concatenate((Tt1.T, Tt2.T),axis =1)
train = np.concatenate((Px.T, Py.T),axis =1)
result.shape
```

Out[37]:

```
(129600, 2)
```

In [38]:

```
x_train = result[0:90720,:]
y_train = train[0:90720,:]
x_test = result[90720:129600,:]
y_test = train[90720:129600,:]
```

In [39]:

```
model=Sequential()
model.add(Dense(512, activation='relu', input_shape=(2,)))
model.add(Dense(512, activation='relu'))
model.add(Dense(2, activation='linear'))
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 dense_3 (Dense)              (None, 512)               1536

 dense_4 (Dense)              (None, 512)               262656

 dense_5 (Dense)              (None, 2)                 1026

=================================================================
Total params: 265,218
Trainable params: 265,218
Non-trainable params: 0
_________________________________________________________________
```

In [40]:

```
model.compile(loss = ['mae'],
              optimizer = 'Adam',
              metrics = ['accuracy'])
```

In [42]:

```
history = model.fit(x_train, y_train, batch_size=128, epochs=50, verbose=1,
validation_data=(x_test,y_test))
```

```
Epoch 1/50
709/709 [==============================] - 8s 11ms/step - loss: 0.4162 - accu
racy: 0.9942 - val_loss: 11.2612 - val_accuracy: 0.9222
Epoch 2/50
709/709 [==============================] - 7s 10ms/step - loss: 0.3756 - accu
racy: 0.9956 - val_loss: 10.6547 - val_accuracy: 0.9270

…

Epoch 48/50
709/709 [==============================] - 7s 10ms/step - loss: 0.2094 - accu
racy: 0.9981 - val_loss: 6.1848 - val_accuracy: 0.9468
Epoch 49/50
709/709 [==============================] - 7s 10ms/step - loss: 0.1991 - accu
racy: 0.9982 - val_loss: 6.1670 - val_accuracy: 0.9456
Epoch 50/50
709/709 [==============================] - 8s 11ms/step - loss: 0.2009 - accu
racy: 0.9981 - val_loss: 6.4274 - val_accuracy: 0.9421
```

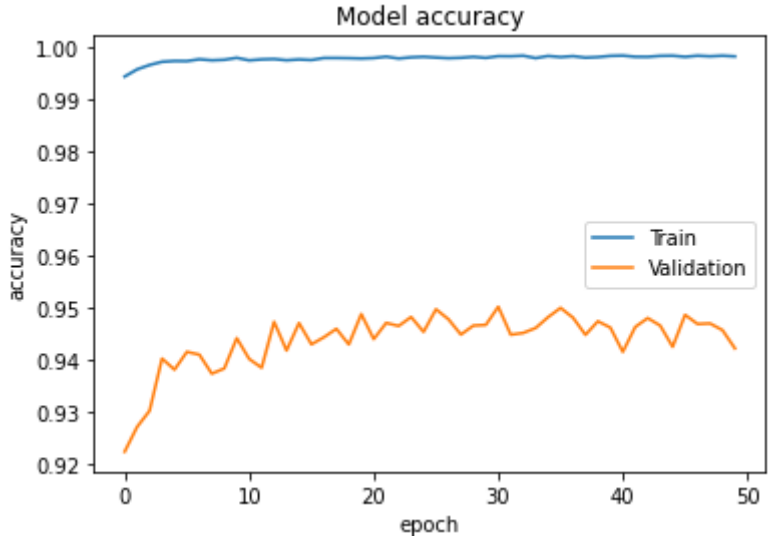
In [43]:

```
Score = model.evaluate(x_test,y_test,verbose = 1)
```

```
1215/1215 [==============================] - 4s 3ms/step - loss: 6.4274 - acc
uracy: 0.9421
```

In [44]:

```
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 ='center right')
plt.show()
```



EXERCISE 5: 3RD DEGREES ROBOTIC ARM

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
```

In [7]:

```
from tensorflow.keras.utils import to_categorical
from keras.models import Sequential
from keras.layers import Dense,Activation,Dropout
from tensorflow.keras.optimizers import RMSprop
from keras.callbacks import EarlyStopping
from sklearn.model_selection import train_test_split
from tensorflow import keras
```

In [8]:

```
l1 = 10
l2 = 40
L3 = 20
px = []
py= []
theta= []
tt1 =[]
tt2 =[]
tt3 =[]
```

In [9]:

```
for i in np.arange(0, 360, 5):
    for j in np.arange(0, 360, 5):
      for k in np.arange(0, 360, 5):
              x = l1*np.cos(np.radians(i)) + l2*np.cos(np.radians(i+j)) +
L3*np.cos(np.radians(i+j+k))
              y = l1*np.sin(np.radians(i)) + l2*np.sin(np.radians(i+j)) +
L3*np.sin(np.radians(i+j+k))
              tt_sum = (i+j+k)%360
              px.append(x)
              py.append(y)
              theta.append(tt_sum)
              tt1.append(i)
              tt2.append(j)
              tt3.append(k)
```

In [10]:

```
Px = np.array([px])
Py = np.array([py])
Tt_sum = np.array([theta])
Tt1= np.array([tt1])
Tt2 = np.array([tt2])
Tt3 = np.array([tt3])
```

In [11]:

```
Px.shape
```

Out[11]:

```
(1, 373248)
```

In [12]:

```
Tt1 = Tt1.astype('float32')
Tt2 = Tt2.astype('float32')
Tt3 = Tt3.astype('float32')
```

In [13]:

```
Tt1 /= 360
Tt2 /= 360
Tt3 /= 360
```

In [14]:

```
result = np.concatenate((Tt1.T, Tt2.T, Tt3.T),axis =1)
train = np.concatenate((Px.T, Py.T, Tt_sum.T),axis =1)
result.shape
```

Out[14]:

```
(373248, 3)
```

In [15]:

```
x_train = result[0:261273,:]
y_train = train[0:261273,:]
x_test = result[261273:373248,:]
y_test = train[261273:373248,:]
```

In [16]:

```
model=Sequential()
model.add(Dense(512, activation='relu', input_shape=(3,)))
model.add(Dense(512, activation='relu'))
model.add(Dense(3, activation='linear'))
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 512)               2048

 dense_1 (Dense)             (None, 512)               262656

 dense_2 (Dense)             (None, 3)                 1539

=================================================================
Total params: 266,243
Trainable params: 266,243
Non-trainable params: 0
_________________________________________________________________
```

In [17]:

```
model.compile(loss = ['mae'],
              optimizer = 'Adam',
              metrics = ['accuracy'])
```

In [19]:

```
history = model.fit(x_train, y_train, batch_size=128, epochs=30, verbose=1,
validation_data=(x_test,y_test))
```

```
Epoch 1/30
```

```
2042/2042 [==============================] - 18s 9ms/step - loss: 7.5767 - ac
curacy: 0.9369 - val_loss: 24.1546 - val_accuracy: 0.9246
Epoch 2/30
2042/2042 [==============================] - 17s 8ms/step - loss: 6.8291 - ac
curacy: 0.9372 - val_loss: 22.8380 - val_accuracy: 0.9253
...
Epoch 29/30
2042/2042 [==============================] - 17s 9ms/step - loss: 2.6042 - ac
curacy: 0.9791 - val_loss: 7.8676 - val_accuracy: 0.9638
Epoch 30/30
2042/2042 [==============================] - 17s 9ms/step - loss: 2.6235 - ac
curacy: 0.9796 - val_loss: 7.6863 - val_accuracy: 0.9698
```

In [20]:

```
Score = model.evaluate(x_test,y_test,verbose = 1)
```

```
3500/3500 [==============================] - 11s 3ms/step - loss: 7.6863 - ac
curacy: 0.9698
```

In [21]:

```
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 ='center right')
plt.show()
```