CNN model using IRIS Dataset

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

In [2]:
```
df=pd.read_csv("C:\\Users\\cheth\\Downloads\\IRIS.csv")
df
```

Out[2]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 5 columns

In [3]:
```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
df['species']=le.fit_transform(df['species'])
df
```

Out[3]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	2
146	6.3	2.5	5.0	1.9	2
147	6.5	3.0	5.2	2.0	2
148	6.2	3.4	5.4	2.3	2
149	5.9	3.0	5.1	1.8	2

150 rows × 5 columns

In [4]:
```
df.isnull().sum()
```

Out[4]:
```
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
dtype: int64
```

In [5]:
```
x=df.drop('species',axis=1)
```

In [6]:
```
y_=df['species']
```

In [7]:
```
import numpy as np

from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder

from keras.models import Sequential
from keras.layers import Dense
from keras.optimizers import Adam
```

In [8]:
```
y1=np.array(y_)
```

In [9]:
```
y2 = y1.reshape(-1, 1)
```

In [10]:
```
encoder = OneHotEncoder(sparse=False)
y = encoder.fit_transform(y2)
```

```
C:\Users\cheth\anaconda3\Lib\site-packages\sklearn\preprocessing\_encoders.py:868: FutureWarning:
`sparse` was renamed to `sparse_output` in version 1.2 and will be removed in 1.4. `sparse_output`
is ignored unless you leave `sparse` to its default value.
  warnings.warn(
```

In [11]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
print('xtrain:',x_train.shape)
print('xtest:',x_test.shape)
print('ytain:',y_train.shape)
print('ytest:',y_test.shape)
```

```
xtrain: (120, 4)
xtest: (30, 4)
ytain: (120, 3)
ytest: (30, 3)
```

In [12]:
```
model = Sequential()
model.add(Dense(10, input_shape=(4,), activation='relu'))
model.add(Dense(10, activation='relu'))
model.add(Dense(3, activation='softmax'))
```

In [13]:
```
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

In [14]:
```
import tensorflow as tf
```

In [15]:
```
tf_callbacks=tf.keras.callbacks.TensorBoard(log_dir='logs/fit',histogram_freq=1)
```

In [18]:

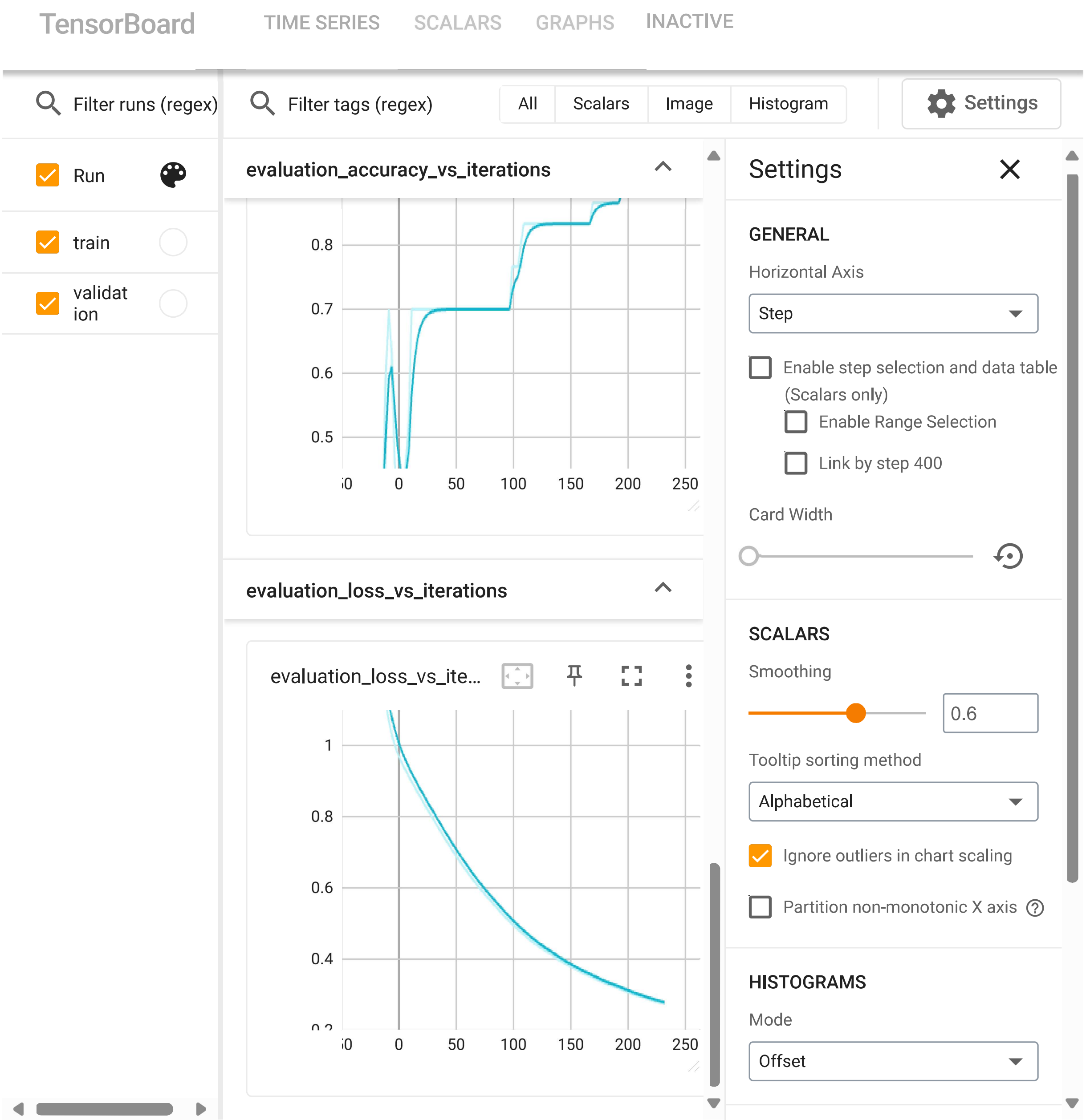
```
history=model.fit(x_train,y_train,validation_data=(x_test,y_test),epochs=50,callbacks=tf_callbacks)
```

```
Epoch 1/50
4/4 [==============================] - 0s 62ms/step - loss: 0.4960 - accuracy: 0.7583 - val_loss:
0.4700 - val_accuracy: 0.8333
Epoch 2/50
4/4 [==============================] - 0s 35ms/step - loss: 0.4889 - accuracy: 0.7833 - val_loss:
0.4631 - val_accuracy: 0.8333
Epoch 3/50
4/4 [==============================] - 0s 33ms/step - loss: 0.4822 - accuracy: 0.8000 - val_loss:
0.4559 - val_accuracy: 0.8333
Epoch 4/50
4/4 [==============================] - 0s 42ms/step - loss: 0.4754 - accuracy: 0.8083 - val_loss:
0.4496 - val_accuracy: 0.8333
Epoch 5/50
4/4 [==============================] - 0s 36ms/step - loss: 0.4681 - accuracy: 0.8250 - val_loss:
0.4438 - val_accuracy: 0.8333
Epoch 6/50
4/4 [==============================] - 0s 38ms/step - loss: 0.4620 - accuracy: 0.8500 - val_loss:
0.4379 - val_accuracy: 0.8333
Epoch 7/50
4/4 [==============================] - 0s 43ms/step - loss: 0.4548 - accuracy: 0.8667 - val_loss:
0.4328 - val_accuracy: 0.8333
Epoch 8/50
4/4 [==============================] - 0s 47ms/step - loss: 0.4485 - accuracy: 0.8667 - val_loss:
0.4274 - val_accuracy: 0.8333
Epoch 9/50
4/4 [==============================] - 0s 50ms/step - loss: 0.4423 - accuracy: 0.8667 - val_loss:
0.4220 - val_accuracy: 0.8333
Epoch 10/50
4/4 [==============================] - 0s 39ms/step - loss: 0.4361 - accuracy: 0.8667 - val_loss:
0.4171 - val_accuracy: 0.8333
Epoch 11/50
4/4 [==============================] - 0s 33ms/step - loss: 0.4305 - accuracy: 0.8667 - val_loss:
0.4117 - val_accuracy: 0.8333
Epoch 12/50
4/4 [==============================] - 0s 33ms/step - loss: 0.4249 - accuracy: 0.8917 - val_loss:
0.4053 - val_accuracy: 0.8333
Epoch 13/50
4/4 [==============================] - 0s 46ms/step - loss: 0.4190 - accuracy: 0.9083 - val_loss:
0.3997 - val_accuracy: 0.8333
Epoch 14/50
4/4 [==============================] - 0s 41ms/step - loss: 0.4138 - accuracy: 0.9167 - val_loss:
0.3943 - val_accuracy: 0.8333
Epoch 15/50
4/4 [==============================] - 0s 60ms/step - loss: 0.4088 - accuracy: 0.9083 - val_loss:
0.3900 - val_accuracy: 0.8333
Epoch 16/50
4/4 [==============================] - 0s 71ms/step - loss: 0.4034 - accuracy: 0.9083 - val_loss:
0.3858 - val_accuracy: 0.8333
Epoch 17/50
4/4 [==============================] - 0s 60ms/step - loss: 0.3987 - accuracy: 0.9083 - val_loss:
0.3811 - val_accuracy: 0.8333
Epoch 18/50
4/4 [==============================] - 0s 60ms/step - loss: 0.3938 - accuracy: 0.9167 - val_loss:
0.3771 - val_accuracy: 0.8333
Epoch 19/50
4/4 [==============================] - 0s 72ms/step - loss: 0.3892 - accuracy: 0.9167 - val_loss:
0.3730 - val_accuracy: 0.8333
Epoch 20/50
4/4 [==============================] - 0s 48ms/step - loss: 0.3845 - accuracy: 0.9167 - val_loss:
0.3688 - val_accuracy: 0.8333
Epoch 21/50
4/4 [==============================] - 0s 57ms/step - loss: 0.3804 - accuracy: 0.9167 - val_loss:
0.3642 - val_accuracy: 0.8333
Epoch 22/50
4/4 [==============================] - 0s 49ms/step - loss: 0.3757 - accuracy: 0.9167 - val_loss:
0.3607 - val_accuracy: 0.8333
Epoch 23/50
4/4 [==============================] - 0s 50ms/step - loss: 0.3717 - accuracy: 0.9167 - val_loss:
0.3580 - val_accuracy: 0.8333
Epoch 24/50
4/4 [==============================] - 0s 49ms/step - loss: 0.3671 - accuracy: 0.9167 - val_loss:
0.3536 - val_accuracy: 0.8333
Epoch 25/50
4/4 [==============================] - 0s 68ms/step - loss: 0.3627 - accuracy: 0.9333 - val_loss:
0.3494 - val_accuracy: 0.8667
Epoch 26/50
```

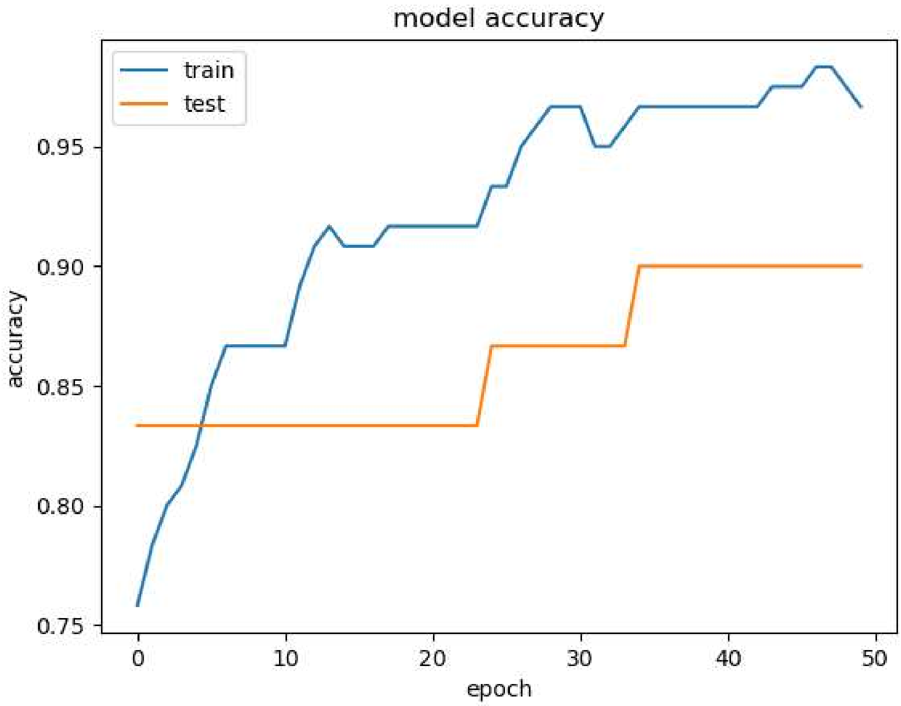
```
4/4 [==============================] - 0s 57ms/step - loss: 0.3584 - accuracy: 0.9333 - val_loss:
0.3454 - val_accuracy: 0.8667
Epoch 27/50
4/4 [==============================] - 0s 45ms/step - loss: 0.3541 - accuracy: 0.9500 - val_loss:
0.3417 - val_accuracy: 0.8667
Epoch 28/50
4/4 [==============================] - 0s 46ms/step - loss: 0.3503 - accuracy: 0.9583 - val_loss:
0.3379 - val_accuracy: 0.8667
Epoch 29/50
4/4 [==============================] - 0s 39ms/step - loss: 0.3465 - accuracy: 0.9667 - val_loss:
0.3340 - val_accuracy: 0.8667
Epoch 30/50
4/4 [==============================] - 0s 39ms/step - loss: 0.3420 - accuracy: 0.9667 - val_loss:
0.3316 - val_accuracy: 0.8667
Epoch 31/50
4/4 [==============================] - 0s 46ms/step - loss: 0.3383 - accuracy: 0.9667 - val_loss:
0.3280 - val_accuracy: 0.8667
Epoch 32/50
4/4 [==============================] - 0s 54ms/step - loss: 0.3349 - accuracy: 0.9500 - val_loss:
0.3266 - val_accuracy: 0.8667
Epoch 33/50
4/4 [==============================] - 0s 37ms/step - loss: 0.3302 - accuracy: 0.9500 - val_loss:
0.3229 - val_accuracy: 0.8667
Epoch 34/50
4/4 [==============================] - 0s 43ms/step - loss: 0.3265 - accuracy: 0.9583 - val_loss:
0.3200 - val_accuracy: 0.8667
Epoch 35/50
4/4 [==============================] - 0s 52ms/step - loss: 0.3223 - accuracy: 0.9667 - val_loss:
0.3156 - val_accuracy: 0.9000
Epoch 36/50
4/4 [==============================] - 0s 39ms/step - loss: 0.3185 - accuracy: 0.9667 - val_loss:
0.3124 - val_accuracy: 0.9000
Epoch 37/50
4/4 [==============================] - 0s 45ms/step - loss: 0.3149 - accuracy: 0.9667 - val_loss:
0.3090 - val_accuracy: 0.9000
Epoch 38/50
4/4 [==============================] - 0s 47ms/step - loss: 0.3120 - accuracy: 0.9667 - val_loss:
0.3068 - val_accuracy: 0.9000
Epoch 39/50
4/4 [==============================] - 0s 39ms/step - loss: 0.3079 - accuracy: 0.9667 - val_loss:
0.3028 - val_accuracy: 0.9000
Epoch 40/50
4/4 [==============================] - 0s 43ms/step - loss: 0.3037 - accuracy: 0.9667 - val_loss:
0.3000 - val_accuracy: 0.9000
Epoch 41/50
4/4 [==============================] - 0s 50ms/step - loss: 0.3001 - accuracy: 0.9667 - val_loss:
0.2976 - val_accuracy: 0.9000
Epoch 42/50
4/4 [==============================] - 0s 60ms/step - loss: 0.2970 - accuracy: 0.9667 - val_loss:
0.2958 - val_accuracy: 0.9000
Epoch 43/50
4/4 [==============================] - 0s 55ms/step - loss: 0.2931 - accuracy: 0.9667 - val_loss:
0.2925 - val_accuracy: 0.9000
Epoch 44/50
4/4 [==============================] - 0s 58ms/step - loss: 0.2894 - accuracy: 0.9750 - val_loss:
0.2897 - val_accuracy: 0.9000
Epoch 45/50
4/4 [==============================] - 0s 58ms/step - loss: 0.2860 - accuracy: 0.9750 - val_loss:
0.2877 - val_accuracy: 0.9000
Epoch 46/50
4/4 [==============================] - 0s 56ms/step - loss: 0.2826 - accuracy: 0.9750 - val_loss:
0.2843 - val_accuracy: 0.9000
Epoch 47/50
4/4 [==============================] - 0s 56ms/step - loss: 0.2791 - accuracy: 0.9833 - val_loss:
0.2817 - val_accuracy: 0.9000
Epoch 48/50
4/4 [==============================] - 0s 67ms/step - loss: 0.2757 - accuracy: 0.9833 - val_loss:
0.2797 - val_accuracy: 0.9000
Epoch 49/50
4/4 [==============================] - 0s 42ms/step - loss: 0.2731 - accuracy: 0.9750 - val_loss:
0.2798 - val_accuracy: 0.9000
Epoch 50/50
4/4 [==============================] - 0s 40ms/step - loss: 0.2693 - accuracy: 0.9667 - val_loss:
0.2759 - val_accuracy: 0.9000
```

In [19]:
```
%reload_ext tensorboard
```

In [20]:
```
tensorboard --logdir logs/fit
```



In [21]:
```
import matplotlib.pyplot as plt
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train','test'],loc='upper left')
plt.show()
```



In [22]:
```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('modelloss')
plt.xlabel('epoch')
plt.ylabel('loss')
plt.legend(['train','test'],loc='upper left')
plt.show()
```