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

from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn import svm
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.preprocessing import StandardScaler

import joblib

import tensorflow.keras as keras
import tensorflow as tf
```

In [ ]:
```
```

In [2]:
```
def make_model(input_shape):
    input_layer = keras.layers.Input(input_shape)

    conv1 = keras.layers.Conv1D(filters=128, kernel_size=3, padding="same")(input_la
    conv1 = keras.layers.BatchNormalization()(conv1)
    conv1 = keras.layers.ReLU()(conv1)

    conv2 = keras.layers.Conv1D(filters=128, kernel_size=3, padding="same")(conv1)
    conv2 = keras.layers.BatchNormalization()(conv2)
    conv2 = keras.layers.ReLU()(conv2)

    conv3 = keras.layers.Conv1D(filters=128, kernel_size=3, padding="same")(conv2)
    conv3 = keras.layers.BatchNormalization()(conv3)
    conv3 = keras.layers.ReLU()(conv3)

    conv4 = keras.layers.Conv1D(filters=128, kernel_size=3, padding="same")(conv3)
    conv4 = keras.layers.BatchNormalization()(conv4)
    conv4 = keras.layers.ReLU()(conv4)

    conv5 = keras.layers.Conv1D(filters=128, kernel_size=3, padding="same")(conv4)
    conv5 = keras.layers.BatchNormalization()(conv5)
    conv5 = keras.layers.ReLU()(conv5)

    gap = keras.layers.GlobalAveragePooling1D()(conv5)




    output_layer = keras.layers.Dense(4, activation="softmax")(gap)

    return keras.models.Model(inputs=input_layer, outputs=output_layer)
```

In [3]:
```
data_s0_air = np.loadtxt('data/matepad_air_0719/1.csv',dtype=str, delimiter=',',skipr
data_s1_air = np.loadtxt('data/matepad_air_0719/4.csv',dtype=str, delimiter=',',skipr
data_s2_air = np.loadtxt('data/matepad_air_0719/5.csv',dtype=str, delimiter=',',skipr
data_s3_air = np.loadtxt('data/matepad_air_0719/7.csv',dtype=str, delimiter=',',skipr
data_s4_air = np.loadtxt('data/matepad_air_0719/8.csv',dtype=str, delimiter=',',skipr

data_s0_pro = np.loadtxt('data/pro/1.csv',dtype=str, delimiter=',',skiprows=1, encod
data_s1_pro = np.loadtxt('data/pro/4.csv',dtype=str, delimiter=',',skiprows=1, encod
data_s2_pro = np.loadtxt('data/pro/5.csv',dtype=str, delimiter=',',skiprows=1, encod
data_s3_pro = np.loadtxt('data/pro/7.csv',dtype=str, delimiter=',',skiprows=1, encod
data_s4_pro = np.loadtxt('data/pro/8.csv',dtype=str, delimiter=',',skiprows=1, encod
data_s3_pro_1 = np.loadtxt('data/pro/横着慢慢走.csv',dtype=str, delimiter=',',skiprov
data_s3_pro_2 = np.loadtxt('data/pro/竖着走.csv',dtype=str, delimiter=',',skiprows=1
data_s3_pro_3 = np.loadtxt('data/pro/退.csv',dtype=str, delimiter=',',skiprows=1, en
data_s0 = np.vstack((data_s0_air,data_s0_pro))
data_s1 = np.vstack((data_s1_air,data_s1_pro))
data_s2 = np.vstack((data_s2_air,data_s2_pro))
data_s3 = np.vstack((data_s3_air,data_s3_pro,data_s3_pro_1,data_s3_pro_2,data_s3_pro_
data_s4 = np.vstack((data_s4_air,data_s4_pro))
print(data_s0.shape)
print(data_s1.shape)
print(data_s2.shape)
print(data_s3.shape)
print(data_s4.shape)
```

```
(129738, 9)
(123228, 9)
(121256, 9)
(263182, 9)
(108195, 9)
```

In [73]:
```
# 1s的采样率
# 也是输入数据的长度
data_sps = 75
axis_num = 9
stride = 20

X = []
y = []

for i in range(0, data_s0.shape[0]-data_sps, stride):
    if data_s0[i:i+data_sps, :].shape[0] == data_sps:
        ## 按行、列不同方式进行reshape
        X.extend(data_s0[i:i+data_sps, :].astype(float).reshape(1, data_sps*axis_num,
        y.append(0)

for i in range(0, data_s1.shape[0]-data_sps, stride):
    if data_s1[i:i+data_sps, :].shape[0] == data_sps:
        ## 按行、列不同方式进行reshape
        X.extend(data_s1[i:i+data_sps, :].astype(float).reshape(1, data_sps*axis_num,
        y.append(1)

for i in range(0, data_s2.shape[0]-data_sps, stride):
    if data_s2[i:i+data_sps, :].shape[0] == data_sps:
        ## 按行、列不同方式进行reshape
        X.extend(data_s2[i:i+data_sps, :].astype(float).reshape(1, data_sps*axis_num,
        y.append(2)

for i in range(0, data_s3.shape[0] - data_sps, stride):
    if data_s3[i:i+data_sps, :].shape[0] == data_sps:
        ## 按行、列不同方式进行reshape
        X.extend(data_s3[i:i+data_sps, :].astype(float).reshape(1, data_sps*axis_num,
        y.append(3)

for i in range(0, data_s4.shape[0] - data_sps, stride):
    if data_s4[i:i+data_sps, :].shape[0] == data_sps:
        ## 按行、列不同方式进行reshape
        X.extend(data_s4[i:i+data_sps, :].astype(float).reshape(1, data_sps*axis_num,
        y.append(3)
```

In [74]:
```
X = np.array(X)
y = np.array(y)
print(X.shape, y.shape)
```

(37264, 675) (37264,)

In [75]:

```
## 切分数据
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1)

print(X_train[0].size)
print(y_train[0])

print(X_train.size)
print(y_train.size)
## 归一化
#scaler = StandardScaler()
#X_train = scaler.fit_transform(X_train.astype(np.float32))
X_train = X_train.reshape((X_train.shape[0], X_train.shape[1], 1))


model = make_model(input_shape=X_train.shape[1:])
```

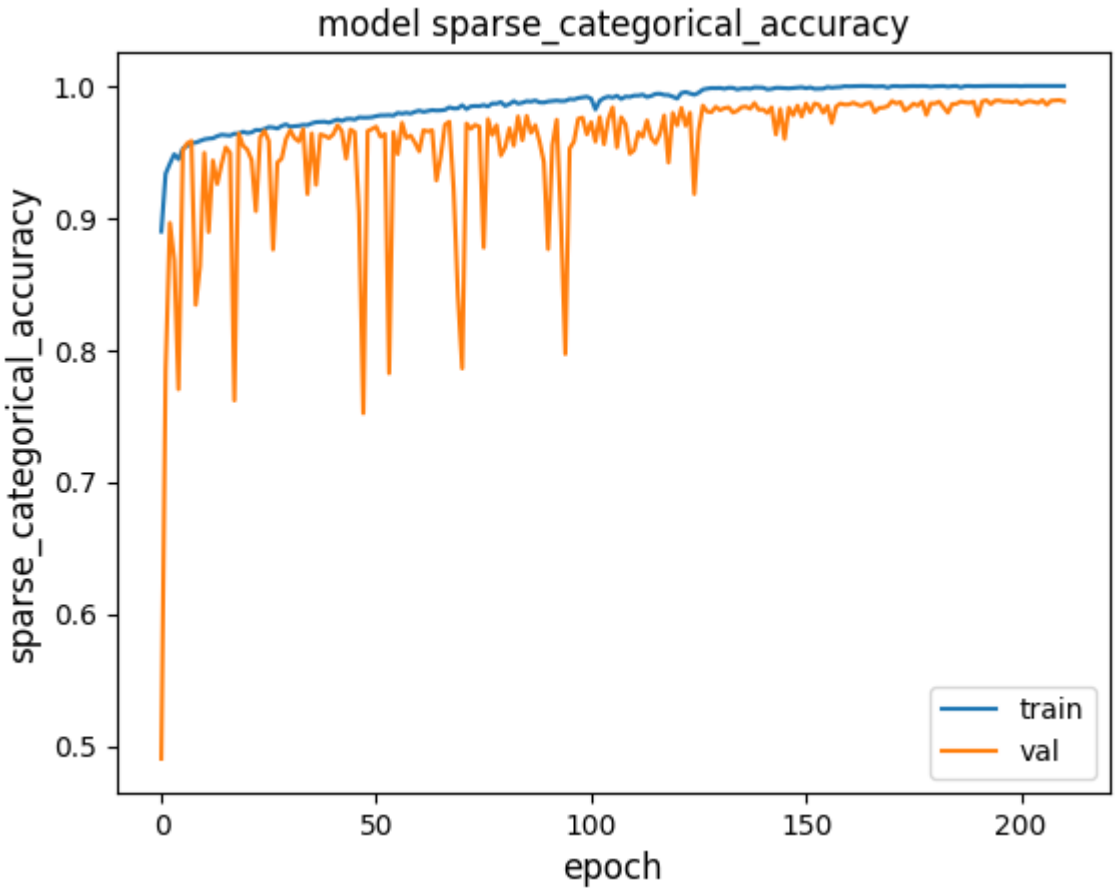
```
675
3
22637475
33537
```

In [76]:
```
epochs = 500
batch_size = 128


callbacks = [
    keras.callbacks.ModelCheckpoint(
        "./500_best_model_d75_out4_stride20_3.h5", save_best_only=True, monitor="val
    ),
    keras.callbacks.ReduceLROnPlateau(
        monitor="val_loss", factor=0.5, patience=20, min_lr=0.00001
    ),
    keras.callbacks.EarlyStopping(monitor="val_loss", patience=40, verbose=1),
]
model.compile(
    optimizer="adam",
    loss="sparse_categorical_crossentropy",
    metrics=["sparse_categorical_accuracy"],
)
history = model.fit(
    X_train,
    y_train,
    batch_size=batch_size,
    epochs=epochs,
    callbacks=callbacks,
    validation_split=0.2,
    verbose=1,
)
```

```
Epoch 1/500
210/210 [==============================] - 24s 104ms/step - loss: 0.3553 - spar
se_categorical_accuracy: 0.8898 - val_loss: 3.0404 - val_sparse_categorical_acc
uracy: 0.4905
Epoch 2/500
210/210 [==============================] - 21s 102ms/step - loss: 0.2209 - spar
se_categorical_accuracy: 0.9335 - val_loss: 0.5783 - val_sparse_categorical_acc
uracy: 0.7853
Epoch 3/500
210/210 [==============================] - 21s 102ms/step - loss: 0.1913 - spar
se_categorical_accuracy: 0.9412 - val_loss: 0.2848 - val_sparse_categorical_acc
uracy: 0.8962
Epoch 4/500
210/210 [==============================] - 22s 103ms/step - loss: 0.1683 - spar
se_categorical_accuracy: 0.9483 - val_loss: 0.3654 - val_sparse_categorical_acc
uracy: 0.8682
Epoch 5/500
210/210 [==============================] - 22s 104ms/step - loss: 0.1742 - spar
se_categorical_accuracy: 0.9445 - val_loss: 0.7827 - val_sparse_categorical_acc
```

In [77]:
```
metric = "sparse_categorical_accuracy"
plt.figure()
plt.plot(history.history[metric])
plt.plot(history.history["val_" + metric])
plt.title("model " + metric)
plt.ylabel(metric, fontsize="large")
plt.xlabel("epoch", fontsize="large")
plt.legend(["train", "val"], loc="best")
plt.show()
```



In [79]:
```
#keras 转换成 tflite
converter = tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model = converter.convert()
open("latest0727-2.tflite", "wb").write(tflite_model)
```

```
INFO:tensorflow:Assets written to: C:\Users\ADMINI~1\AppData\Local\Temp\tmp94y6p11u\assets

INFO:tensorflow:Assets written to: C:\Users\ADMINI~1\AppData\Local\Temp\tmp94y6p11u\assets
```

Out[79]: 803348

In [ ]: