Code used to train and test the K-nn classifier.

In [40]:

```
import pickle
import os
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.metrics import recall_score
from sklearn.metrics import precision_score
from sklearn.metrics import f1_score
from sklearn.neighbors import KNeighborsClassifier
```

In [49]:

```
#load data
path_to_file = "HDataset_norm.pkl"
infile_c = open(path_to_file,'rb')
HDataset = pickle.load(infile_c)
infile_c.close()
```

In [42]:

```
# Split dataset
X_train, X_test, y_train, y_test = train_test_split(HDataset["vectors"], HDataset["label
```

In [43]:

```
# Train Classifier
nbrs = KNeighborsClassifier(n_neighbors=5).fit(X_train,y_train)
```

In [44]:

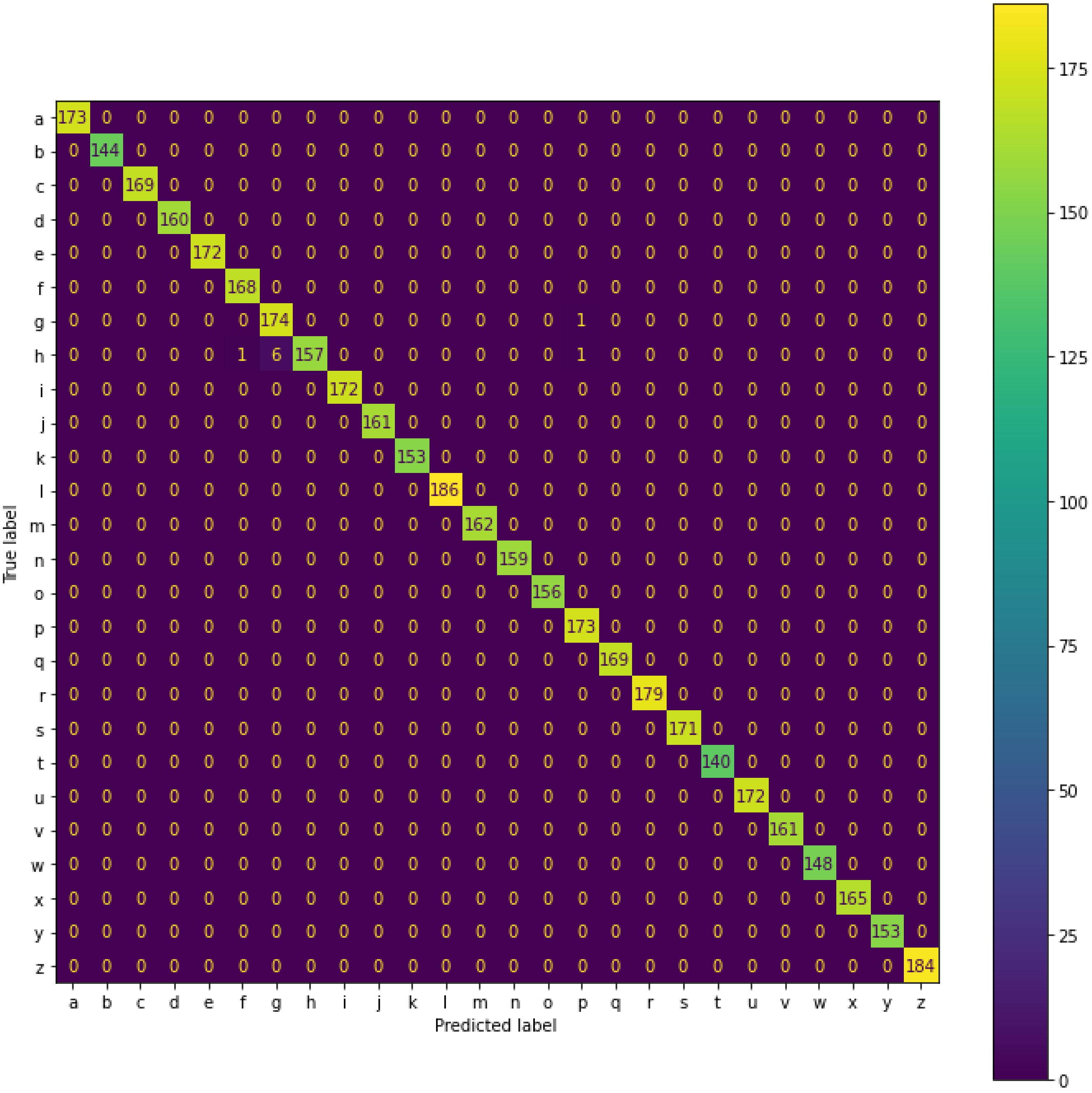
```
# Prediction on Test set
y_pred = nbrs.predict(X_test)
```

In [45]:

```
from sklearn.metrics import confusion_matrix,ConfusionMatrixDisplay
import string

cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=list(string.ascii_lowe

fig, ax = plt.subplots(figsize=(12,12))
disp.plot(ax=ax)
plt.show()
```



	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a	173	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
b	0	144	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c	0	0	169	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
d	0	0	0	160	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
e	0	0	0	0	172	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
f	0	0	0	0	0	168	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
g	0	0	0	0	0	0	174	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
h	0	0	0	0	0	1	6	157	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
i	0	0	0	0	0	0	0	0	172	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
j	0	0	0	0	0	0	0	0	0	161	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
k	0	0	0	0	0	0	0	0	0	0	153	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
l	0	0	0	0	0	0	0	0	0	0	0	186	0	0	0	0	0	0	0	0	0	0	0	0	0	0
m	0	0	0	0	0	0	0	0	0	0	0	0	162	0	0	0	0	0	0	0	0	0	0	0	0	0
n	0	0	0	0	0	0	0	0	0	0	0	0	0	159	0	0	0	0	0	0	0	0	0	0	0	0
o	0	0	0	0	0	0	0	0	0	0	0	0	0	0	156	0	0	0	0	0	0	0	0	0	0	0
p	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	173	0	0	0	0	0	0	0	0	0	0
q	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	169	0	0	0	0	0	0	0	0	0
r	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	179	0	0	0	0	0	0	0	0
s	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	171	0	0	0	0	0	0	0
t	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	140	0	0	0	0	0	0
u	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	172	0	0	0	0	0
v	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	161	0	0	0	0
w	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	148	0	0	0
x	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	165	0	0
y	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	153	0
z	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	184

In [47]:

```
print("Accuracy: ", accuracy_score(y_test, y_pred))
print("Recal: ",recall_score(y_test, y_pred,average='macro'))
print("Precision: ",precision_score(y_test, y_pred,average='macro'))
print("F1: ",f1_score(y_test, y_pred,average='macro'))
```

```
Accuracy:  0.9979020979020979
Recal:  0.9979154179154179
Precision:  0.9980508052105684
F1:  0.9979508647752586
```

In [ ]: