In [ ]:

```
import pandas as pd
import numpy as np
import random
import matplotlib.pyplot as plt
from tensorflow.keras.utils import to_categorical

from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Dense, Flatten, Dropout
```

In [ ]:

```
train = pd.read_csv('sign_mnist_train.csv')
test = pd.read_csv('sign_mnist_test.csv')
```

In [ ]:

```
#Datasets as numpy arrays
train_data = np.array(train, dtype = 'float32')
test_data = np.array(test, dtype='float32')
```

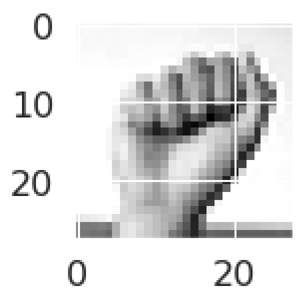
In [ ]:

```
#Define class labels for easy interpretation
class_names = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'K', 'L', 'M',
               'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y' ]
```

In [ ]:

```
#Sanity check - plot a few images and labels
i = random.randint(1,train.shape[0])
fig1, ax1 = plt.subplots(figsize=(2,2))
plt.imshow(train_data[i,1:].reshape((28,28)), cmap='gray')
print("Label for the image is: ", class_names[int(train_data[i,0])])
```

```
Label for the image is:  A
```

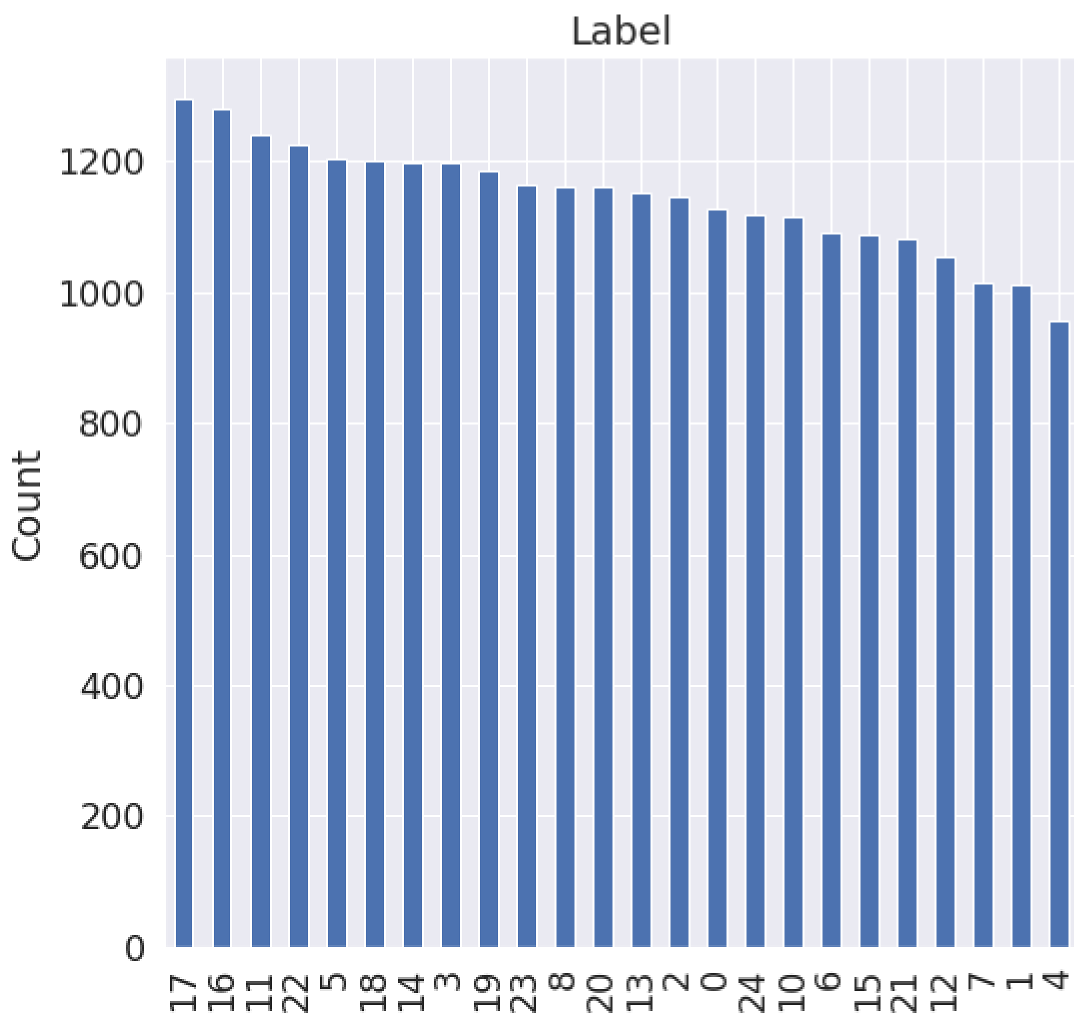


In [ ]:

```
# Data distribution visualization
fig = plt.figure(figsize=(18,18))
ax1 = fig.add_subplot(221)
train['label'].value_counts().plot(kind='bar', ax=ax1)
ax1.set_ylabel('Count')
ax1.set_title('Label')
```

Out[44]:

```
Text(0.5, 1.0, 'Label')
```



In [ ]:
```
#Dataset seems to be fairly balanced.

#Normalize / scale X values
X_train = train_data[:, 1:] /255.
X_test = test_data[:, 1:] /255.
```

In [ ]:
```
#Convert y to categorical if planning on using categorical cross entropy
#No need to do this if using sparse categorical cross entropy
y_train = train_data[:, 0]
y_train_cat = to_categorical(y_train, num_classes=25)
```

In [ ]:
```
y_test = test_data[:,0]
y_test_cat = to_categorical(y_test, num_classes=25)
```

In [ ]:
```
#Reshape for the neural network
X_train = X_train.reshape(X_train.shape[0], *(28, 28, 1))
X_test = X_test.reshape(X_test.shape[0], *(28, 28, 1))
```

In [ ]:
```
#Model

model = Sequential()

model.add(Conv2D(32, (3, 3), input_shape = (28,28,1), activation='relu'))
model.add(MaxPooling2D(pool_size = (2, 2)))
model.add(Dropout(0.2))

model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size = (2, 2)))
model.add(Dropout(0.2))

model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size = (2, 2)))
model.add(Dropout(0.2))

model.add(Flatten())

model.add(Dense(128, activation = 'relu'))
model.add(Dense(25, activation = 'softmax'))
```

In [ ]:

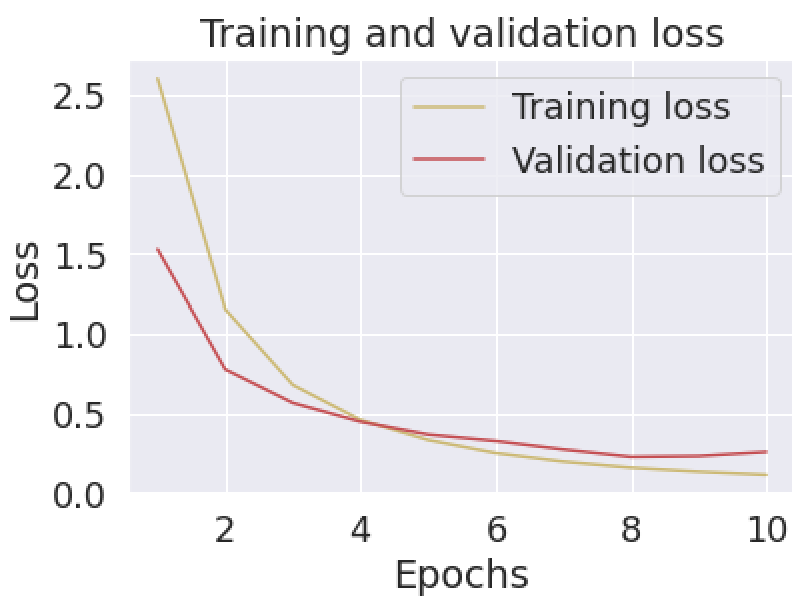
```
#If your targets are one-hot encoded, use categorical_crossentropy. Examples
# If your targets are integers, use sparse_categorical_crossentropy.

#model.compile(loss ='sparse_categorical_crossentropy', optimizer='adam', met
model.compile(loss ='categorical_crossentropy', optimizer='adam', metrics =['
model.summary()

#history = model.fit(X_train, y_train, batch_size = 128, epochs = 10, verbose
history = model.fit(X_train, y_train_cat, batch_size = 128, epochs = 10, verb
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_3 (Conv2D)           (None, 26, 26, 32)        320

 max_pooling2d_3 (MaxPooling  (None, 13, 13, 32)       0
 2D)

 dropout_3 (Dropout)         (None, 13, 13, 32)        0

 conv2d_4 (Conv2D)           (None, 11, 11, 64)        18496

 max_pooling2d_4 (MaxPooling  (None, 5, 5, 64)         0
 2D)

 dropout_4 (Dropout)         (None, 5, 5, 64)          0

 conv2d_5 (Conv2D)           (None, 3, 3, 128)         73856

 max_pooling2d_5 (MaxPooling  (None, 1, 1, 128)        0
 2D)

 dropout_5 (Dropout)         (None, 1, 1, 128)         0

 flatten_1 (Flatten)         (None, 128)               0

 dense_2 (Dense)             (None, 128)               16512

 dense_3 (Dense)             (None, 25)                3225

=================================================================
Total params: 112,409
Trainable params: 112,409
Non-trainable params: 0
_________________________________________________________________
Epoch 1/10
215/215 [==============================] - 20s 92ms/step - loss: 2.6056 -
acc: 0.1990 - val_loss: 1.5294 - val_acc: 0.4908
Epoch 2/10
215/215 [==============================] - 20s 93ms/step - loss: 1.1556 -
acc: 0.6103 - val_loss: 0.7779 - val_acc: 0.7490
Epoch 3/10
215/215 [==============================] - 20s 92ms/step - loss: 0.6796 -
acc: 0.7689 - val_loss: 0.5665 - val_acc: 0.8098
```

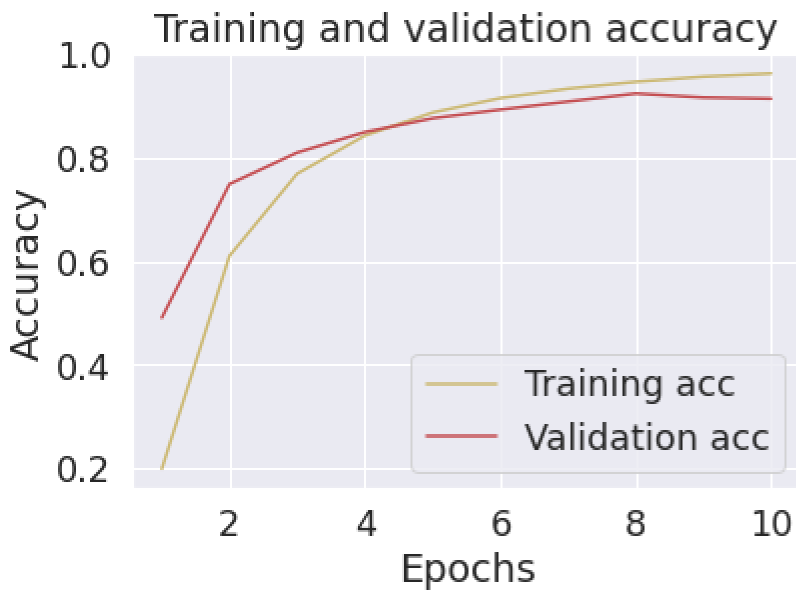
```
Epoch 4/10
215/215 [==============================] - 20s 92ms/step - loss: 0.4597 -
acc: 0.8424 - val_loss: 0.4485 - val_acc: 0.8493
Epoch 5/10
215/215 [==============================] - 20s 92ms/step - loss: 0.3329 -
acc: 0.8873 - val_loss: 0.3674 - val_acc: 0.8762
Epoch 6/10
215/215 [==============================] - 20s 92ms/step - loss: 0.2505 -
acc: 0.9151 - val_loss: 0.3264 - val_acc: 0.8928
Epoch 7/10
215/215 [==============================] - 20s 92ms/step - loss: 0.1965 -
acc: 0.9333 - val_loss: 0.2731 - val_acc: 0.9084
Epoch 8/10
215/215 [==============================] - 20s 92ms/step - loss: 0.1583 -
acc: 0.9465 - val_loss: 0.2265 - val_acc: 0.9239
Epoch 9/10
215/215 [==============================] - 20s 92ms/step - loss: 0.1326 -
acc: 0.9565 - val_loss: 0.2314 - val_acc: 0.9162
Epoch 10/10
215/215 [==============================] - 20s 92ms/step - loss: 0.1130 -
acc: 0.9626 - val_loss: 0.2577 - val_acc: 0.9140
```

In [ ]:

```
#plot the training and validation accuracy and loss at each epoch
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(loss) + 1)
plt.plot(epochs, loss, 'y', label='Training loss')
plt.plot(epochs, val_loss, 'r', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```



In [ ]:
```
acc = history.history['acc']
val_acc = history.history['val_acc']

plt.plot(epochs, acc, 'y', label='Training acc')
plt.plot(epochs, val_acc, 'r', label='Validation acc')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```



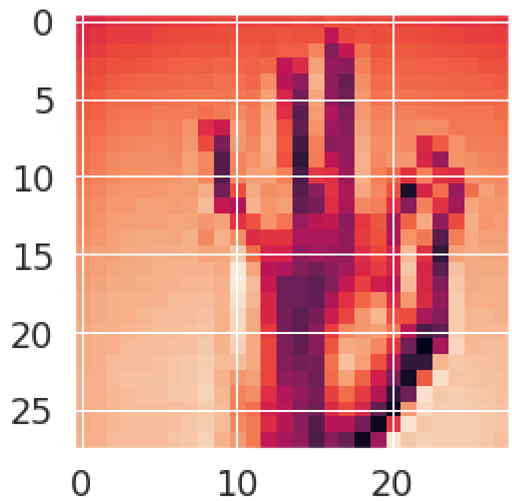
In [ ]:
```
predict_x=model.predict(X_test)
classes_x=np.argmax(predict_x,axis=1)
```

In [ ]:
```
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test, classes_x)
print('Accuracy Score = ', accuracy)
```

```
Accuracy Score =  0.9139709983268266
```

In [ ]:
```
i = random.randint(1,len(classes_x))
plt.imshow(X_test[i,:,:,0])
print("Predicted Label: ", class_names[int(classes_x[i])])
print("True Label: ", class_names[int(y_test[i])])
```

```
Predicted Label:  F
True Label:  F
```

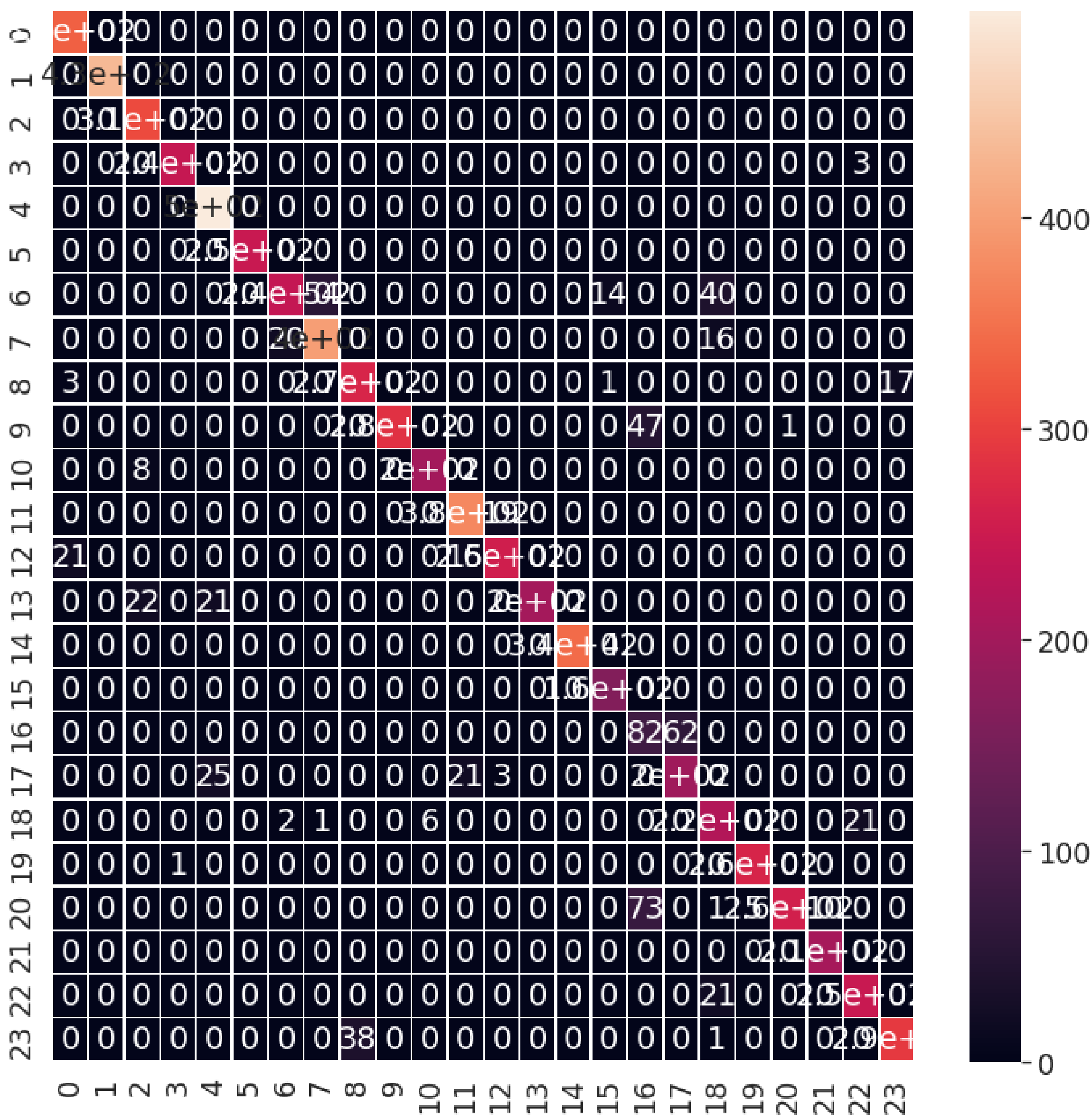


In [ ]:

```
from sklearn.metrics import confusion_matrix
import seaborn as sns
#Print confusion matrix
cm = confusion_matrix(y_test, classes_x)

fig, ax = plt.subplots(figsize=(12,12))
sns.set(font_scale=1.6)
sns.heatmap(cm, annot=True, linewidths=.5, ax=ax)
```

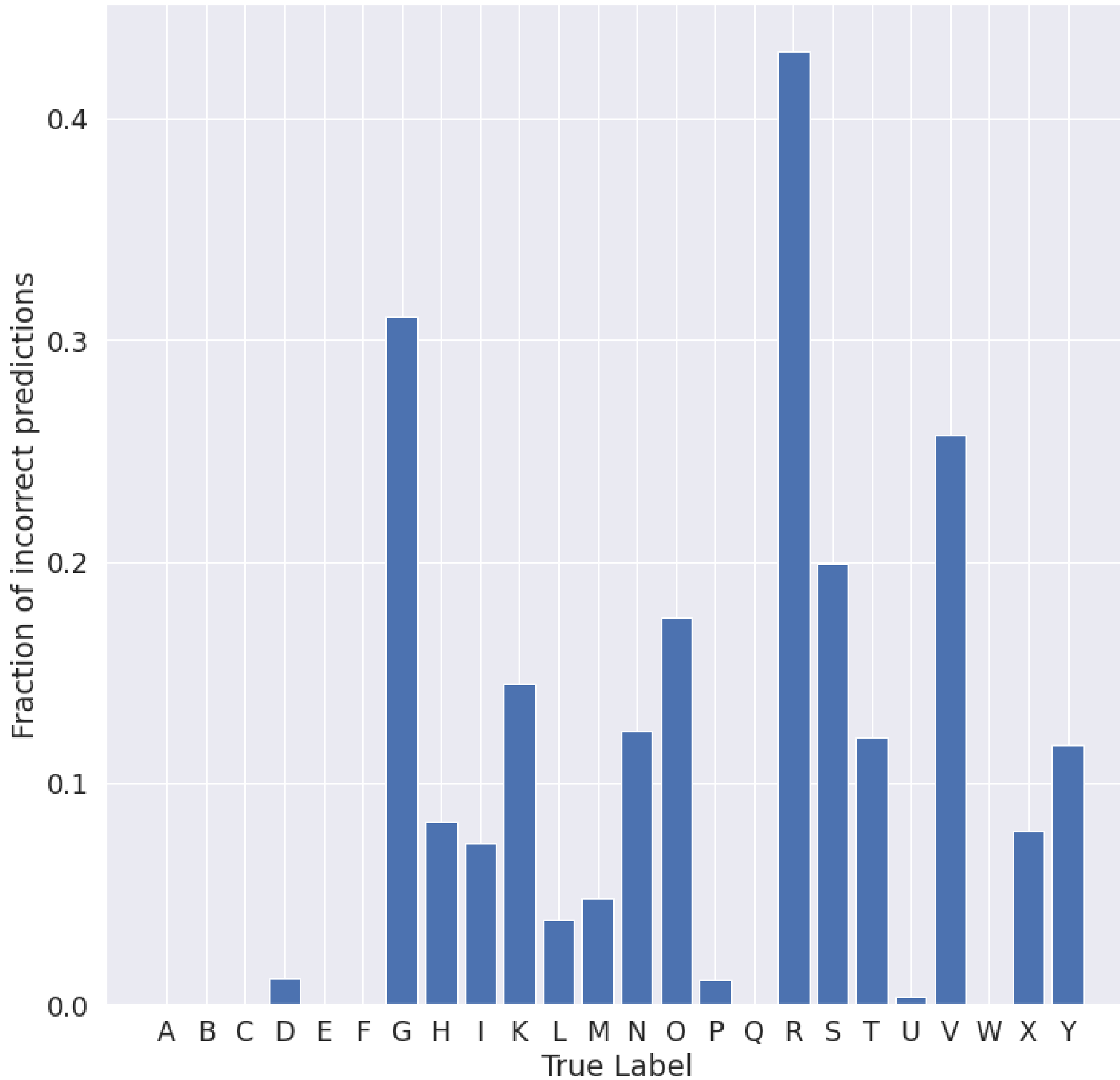
Out[56]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f446bf1b7d0>
```



In [ ]:
```
#PLot fractional incorrect misclassifications
incorr_fraction = 1 - np.diag(cm) / np.sum(cm, axis=1)
fig, ax = plt.subplots(figsize=(12,12))
plt.bar(np.arange(24), incorr_fraction)
plt.xlabel('True Label')
plt.ylabel('Fraction of incorrect predictions')
plt.xticks(np.arange(24), class_names)
```

Out[57]:
```
([<matplotlib.axis.XTick at 0x7f446f9f0b90>,
  <matplotlib.axis.XTick at 0x7f446f9f0f90>,
  <matplotlib.axis.XTick at 0x7f446f9f0490>,
  <matplotlib.axis.XTick at 0x7f446f903550>,
  <matplotlib.axis.XTick at 0x7f446f903a90>,
  <matplotlib.axis.XTick at 0x7f446f903910>,
  <matplotlib.axis.XTick at 0x7f446f90f3d0>,
  <matplotlib.axis.XTick at 0x7f446f90f310>,
  <matplotlib.axis.XTick at 0x7f446f90f7d0>,
  <matplotlib.axis.XTick at 0x7f446f91f3d0>,
  <matplotlib.axis.XTick at 0x7f446f91f2d0>,
  <matplotlib.axis.XTick at 0x7f446f91f590>,
  <matplotlib.axis.XTick at 0x7f446f930390>,
  <matplotlib.axis.XTick at 0x7f446f91f450>,
  <matplotlib.axis.XTick at 0x7f446f90f5d0>,
  <matplotlib.axis.XTick at 0x7f446f930790>,
  <matplotlib.axis.XTick at 0x7f446f930c10>,
  <matplotlib.axis.XTick at 0x7f446f8d01d0>,
  <matplotlib.axis.XTick at 0x7f446f8d06d0>,
  <matplotlib.axis.XTick at 0x7f446f8d0590>,
  <matplotlib.axis.XTick at 0x7f446f8d71d0>,
  <matplotlib.axis.XTick at 0x7f446f8d76d0>,
  <matplotlib.axis.XTick at 0x7f446f8d7590>,
  <matplotlib.axis.XTick at 0x7f446f8d7bd0>],
 [Text(0, 0, 'A'),
  Text(0, 0, 'B'),
  Text(0, 0, 'C'),
  Text(0, 0, 'D'),
  Text(0, 0, 'E'),
  Text(0, 0, 'F'),
  Text(0, 0, 'G'),
  Text(0, 0, 'H'),
  Text(0, 0, 'I'),
  Text(0, 0, 'K'),
  Text(0, 0, 'L'),
  Text(0, 0, 'M'),
  Text(0, 0, 'N'),
  Text(0, 0, 'O'),
  Text(0, 0, 'P'),
  Text(0, 0, 'Q'),
  Text(0, 0, 'R'),
  Text(0, 0, 'S'),
  Text(0, 0, 'T'),
  Text(0, 0, 'U'),
  Text(0, 0, 'V'),
  Text(0, 0, 'W'),
  Text(0, 0, 'X'),
  Text(0, 0, 'Y')])
```


Fraction of incorrect predictions
0.0
0.1
0.2
0.3
0.4
A B C D E F G H I K L M N O P Q R S T U V W X Y
True Label

In [ ]: