In [3]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten
```

```
WARNING:tensorflow:From C:\Users\saksh\dsml27F\envs\dsml27_env1\Lib\site
-packages\keras\src\losses.py:2976: The name tf.losses.sparse_softmax_cr
oss_entropy is deprecated. Please use tf.compat.v1.losses.sparse_softmax
_cross_entropy instead.
```

In [4]:

```
data = pd.read_csv('../../dataset/A_Z Handwritten Data.csv (1).zip')
```

In [6]:

```
data.head()
```

Out[6]:

	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	...	0.639	0.640	0.641	0.642	0.643	0.644
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0

5 rows × 785 columns

In [7]:

```
X = data.drop('0', axis=1)
y = data['0']
```

In [8]:

```
X.shape
```

Out[8]:

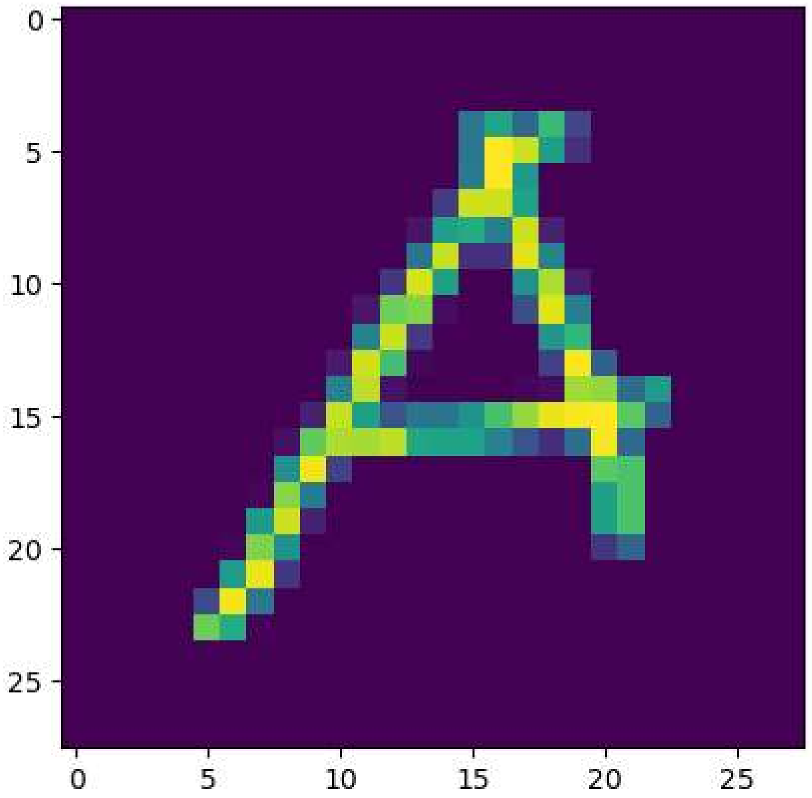
```
(372450, 784)
```

In [9]:

```
img = X.iloc[3,:].values.reshape(28,28)
plt.imshow(img)
```

Out[9]:

```
<matplotlib.image.AxesImage at 0x2c457462dd0>
```



In [10]:

```
X = X/255
```

In [11]:

```
from tensorflow.keras.utils import to_categorical
```

In [12]:

```
ya = to_categorical(y,num_classes=26)
ya.shape
```

Out[12]:

```
(372450, 26)
```

Model building

In [13]:

```
from tensorflow.keras.layers import Dense
from tensorflow.keras.models import Sequential
```

In [14]:

```
model = Sequential()
model.add(Dense(128,activation='relu',input_shape=(784,)))
model.add(Dense(64,activation='relu'))
model.add(Dense(26,activation='softmax'))

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

```
WARNING:tensorflow:From C:\Users\saksh\dsml27F\envs\dsml27_env1\Lib\site
-packages\keras\src\backend.py:873: The name tf.get_default_graph is dep
recated. Please use tf.compat.v1.get_default_graph instead.

WARNING:tensorflow:From C:\Users\saksh\dsml27F\envs\dsml27_env1\Lib\site
-packages\keras\src\optimizers\__init__.py:309: The name tf.train.Optimi
zer is deprecated. Please use tf.compat.v1.train.Optimizer instead.
```

In [15]:

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 dense (Dense)               (None, 128)               100480    
                                                                 
 dense_1 (Dense)             (None, 64)                8256      
                                                                 
 dense_2 (Dense)             (None, 26)                1690      
                                                                 
=================================================================
Total params: 110426 (431.35 KB)
Trainable params: 110426 (431.35 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [16]:

```
model.fit(X,ya,batch_size=32,epochs=10)
```

```
Epoch 1/10
WARNING:tensorflow:From C:\Users\saksh\dsml27F\envs\dsml27_env1\Lib\site
-packages\keras\src\utils\tf_utils.py:492: The name tf.ragged.RaggedTens
orValue is deprecated. Please use tf.compat.v1.ragged.RaggedTensorValue
instead.

WARNING:tensorflow:From C:\Users\saksh\dsml27F\envs\dsml27_env1\Lib\site
-packages\keras\src\engine\base_layer_utils.py:384: The name tf.executin
g_eagerly_outside_functions is deprecated. Please use tf.compat.v1.execu
ting_eagerly_outside_functions instead.

11640/11640 [==============================] - 136s 11ms/step - loss: 0.
2424 - accuracy: 0.9337
Epoch 2/10
11640/11640 [==============================] - 131s 11ms/step - loss: 0.
1402 - accuracy: 0.9640
Epoch 3/10
11640/11640 [==============================] - 132s 11ms/step - loss: 0.
1344 - accuracy: 0.9678
Epoch 4/10
11640/11640 [==============================] - 130s 11ms/step - loss: 0.
1371 - accuracy: 0.9686
Epoch 5/10
11640/11640 [==============================] - 133s 11ms/step - loss: 0.
1397 - accuracy: 0.9694
Epoch 6/10
11640/11640 [==============================] - 130s 11ms/step - loss: 0.
1399 - accuracy: 0.9708
Epoch 7/10
11640/11640 [==============================] - 131s 11ms/step - loss: 0.
1465 - accuracy: 0.9706
Epoch 8/10
11640/11640 [==============================] - 130s 11ms/step - loss: 0.
1472 - accuracy: 0.9712
Epoch 9/10
11640/11640 [==============================] - 131s 11ms/step - loss: 0.
1476 - accuracy: 0.9717
Epoch 10/10
11640/11640 [==============================] - 130s 11ms/step - loss: 0.
1500 - accuracy: 0.9725
```

Out[16]:

```
<keras.src.callbacks.History at 0x2c4c044ac10>
```

In [18]:

```
y[:10]
```

Out[18]:

```
0    0
1    0
2    0
3    0
4    0
5    0
6    0
7    0
8    0
9    0
Name: 0, dtype: int64
```

In [25]:

```
model.predict_on_batch(X.iloc[:5,:]).argmax(axis=1)
```

Out[25]:

```
array([0, 0, 0, 0, 0], dtype=int64)
```

In [26]:

```
img = X.iloc[0,:].values.reshape(1,784)
model.predict_on_batch(img).argmax()
```

Out[26]:

```
0
```

Predictions on own handwriting

In [27]:

```
import cv2
```

In [28]:

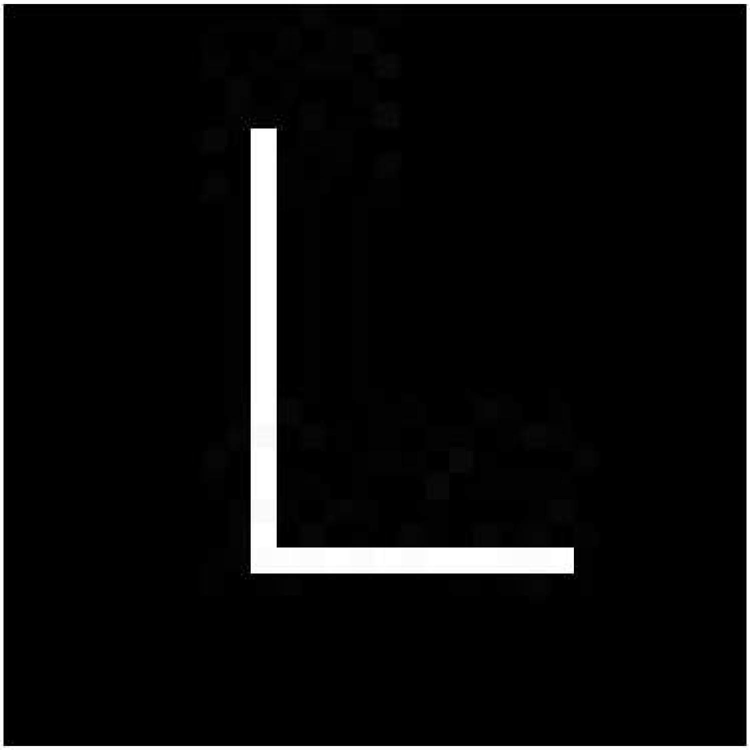
```
path = '../../dataset/LL.jpg'
```

In [30]:

```
image = cv2.imread(path)
```

In [31]:

```
plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
plt.axis('off')
plt.show()
```



In [70]:

```
def get_alphabet(filename):
    path = '../../dataset/Alphabet Recognization/' + filename
    A = cv2.imread(path,0)
    A = cv2.resize(A,(30,30))
    A = A/255
    A = A.reshape(1,784)
    return model.predict_on_batch(A).argmax()
```

In [44]:

```
get_alphabet('LL.jpg')
```

Out[44]:

```
20
```

In [45]:

```
import os
```

In [48]:

```
for file in filenames:
    if os.path.exists(file):
        image = cv2.imread(file)
```

In [71]:

```
def get_alphabet(filename):
    path = '../../dataset/Alphabet Recognization/' + filename
    A = cv2.imread(path,0)
    A = cv2.resize(A,(30,30))
    A = A/255
    A = A.reshape(1,784)
    return model.predict_on_batch(A).argmax()
```

In [64]:

```
get_alphabet('B.jpg')
```

Out[64]:

```
16
```

In [65]:

```
import os
```

In [66]:

```
filenames = os.listdir('../../dataset/Alphabet Recognization/')
filenames
```

Out[66]:

```
['A.jpg',
 'AA.jpg',
 'B.jpg',
 'BB.jpg',
 'C.jpg',
 'CC.jpg',
 'D.jpg',
 'DD.jpg',
 'E.jpg',
 'EE.jpg',
 'F.jpg',
 'FF.jpg',
 'G.jpg',
 'GG.jpg',
 'H.jpg',
 'HH.jpg',
 'I.jpg',
 'II.jpg',
 'J.jpg',
 'JJ.jpg',
 'K.jpg',
 'KK (1).jpg',
 'L.jpg',
 'LL.jpg',
 'M.jpg',
 'MM.jpg',
 'N.jpg',
 'NN.jpg',
 'O.jpg',
 'OO.jpg',
 'P.jpg',
 'PP.jpg',
 'Q.jpg',
 'QQ.jpg',
 'R.jpg',
 'RR.jpg',
 'S.jpg',
 'SS.jpg',
 'T.jpg',
 'TT.jpg',
 'U.jpg',
 'UU.jpg',
 'V.jpg',
 'VV.jpg',
 'W.jpg',
 'WW.jpg',
 'X.jpg',
 'XX.jpg',
 'Y.jpg',
 'YY.jpg',
 'Z.jpg',
 'ZZ.jpg']
```

In [67]:

```
for file in filenames:
    yp = get_alphabet(file)
    print(file,'\t', yp)
```

```
A.jpg    0
AA.jpg   0
B.jpg    16
BB.jpg   1
C.jpg    2
CC.jpg   2
D.jpg    3
DD.jpg   3
E.jpg    4
EE.jpg   4
F.jpg    17
FF.jpg   4
G.jpg    16
GG.jpg   18
H.jpg    7
HH.jpg   4
I.jpg    8
II.jpg   8
J.jpg    8
JJ.jpg   9
K.jpg    10
KK (1).jpg       10
L.jpg    12
LL.jpg   20
M.jpg    12
MM.jpg   12
N.jpg    22
NN.jpg   13
O.jpg    3
OO.jpg   16
P.jpg    15
PP.jpg   15
Q.jpg    16
QQ.jpg   16
R.jpg    16
RR.jpg   7
S.jpg    6
SS.jpg   7
T.jpg    24
TT.jpg   24
U.jpg    20
UU.jpg   20
V.jpg    14
VV.jpg   20
W.jpg    22
WW.jpg   22
X.jpg    9
XX.jpg   23
Y.jpg    24
YY.jpg   24
Z.jpg    25
ZZ.jpg   25
```

In [101]:

```
import os

predicted_class_indices = {
    'A.jpg': 0, 'AA.jpg': 0, 'B.jpg': 16, 'BB.jpg': 1, 'C.jpg': 2, 'CC.jpg
    'E.jpg': 4, 'EE.jpg': 4, 'F.jpg': 17, 'FF.jpg': 4, 'G.jpg': 16, 'GG.jp
    'I.jpg': 8, 'II.jpg': 8, 'J.jpg': 8, 'JJ.jpg': 9, 'K.jpg': 10, 'KK (1)
    'M.jpg': 12, 'MM.jpg': 12, 'N.jpg': 22, 'NN.jpg': 13, 'O.jpg': 3, 'OO.
    'Q.jpg': 16, 'QQ.jpg': 16, 'R.jpg': 16, 'RR.jpg': 7, 'S.jpg': 6, 'SS.j
    'U.jpg': 20, 'UU.jpg': 20, 'V.jpg': 14, 'VV.jpg': 20, 'W.jpg': 22, 'WW
    'Y.jpg': 24, 'YY.jpg': 24, 'Z.jpg': 25, 'ZZ.jpg': 25
}

# Ground truth labels
ground_truth_labels = {
    'A.jpg': 0, 'AA.jpg': 0, 'B.jpg': 1, 'BB.jpg': 1, 'C.jpg': 2, 'CC.jpg
    'E.jpg': 4, 'EE.jpg': 4, 'F.jpg': 5, 'FF.jpg': 5, 'G.jpg': 6, 'GG.jpg
    'I.jpg': 8, 'II.jpg': 8, 'J.jpg': 9, 'JJ.jpg': 9, 'K.jpg': 10, 'KK (1)
    'M.jpg': 12, 'MM.jpg': 12, 'N.jpg': 13, 'NN.jpg': 13, 'O.jpg': 14, 'OO
    'Q.jpg': 16, 'QQ.jpg': 16, 'R.jpg': 17, 'RR.jpg': 17, 'S.jpg': 18, 'SS
    'U.jpg': 20, 'UU.jpg': 20, 'V.jpg': 21, 'VV.jpg': 21, 'W.jpg': 22, 'WW
    'Y.jpg': 24, 'YY.jpg': 24, 'Z.jpg': 25, 'ZZ.jpg': 25
}

# Calculate accuracy
correct_predictions = sum(1 for file, predicted_class in predicted_class_i
total_images = len(predicted_class_indices)
accuracy = (correct_predictions / total_images) * 100 if total_images > 0

print(f"\nSummary:")
print(f"Correctly Predicted: {correct_predictions}/{total_images}")
print(f"Accuracy: {accuracy:.2f}%")
```

```

Summary:
Correctly Predicted: 31/52
Accuracy: 59.62%
```

In [ ]: