digit-detection-using-cnn

May 4, 2023

Importing data from Kaggle

[1]:
```
! pip install -q kaggle
from google.colab import files
files.upload()
! mkdir ~/.kaggle
! cp kaggle.json ~/.kaggle/
! chmod 600 ~/.kaggle/kaggle.json
```

```
<IPython.core.display.HTML object>
```

```
Saving kaggle.json to kaggle (2).json
mkdir: cannot create directory ‘/root/.kaggle’: File exists
```

[2]:
```
!kaggle datasets download -d devanshusingh/
  ↪hand-written-digits-for-object-detection
```

```
hand-written-digits-for-object-detection.zip: Skipping, found more recently
modified local copy (use --force to force download)
```

[ ]:
```
!unzip hand-written-digits-for-object-detection.zip
```

Importing Dependencies

[4]:
```
import os
import cv2 as cv
import keras
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib.patches as patches
from sklearn.model_selection import train_test_split
from keras.layers import Conv2D, MaxPool2D, Dense, GlobalAveragePooling2D, Input
from keras.models import Sequential, Model
from sklearn.metrics import mean_absolute_error
```

[5]:
```
df = pd.read_csv('/content/train_label.csv')
```

[6]:
```
df.head()
```

[6]:

	image_path	label	x_mark	y_mark	width	height
0	0.png	5	23.0	0.0	51.0	28.0
1	1.png	0	43.0	34.0	71.0	62.0
2	2.png	4	43.0	23.0	71.0	51.0
3	3.png	1	3.0	27.0	31.0	55.0
4	4.png	9	32.0	12.0	60.0	40.0

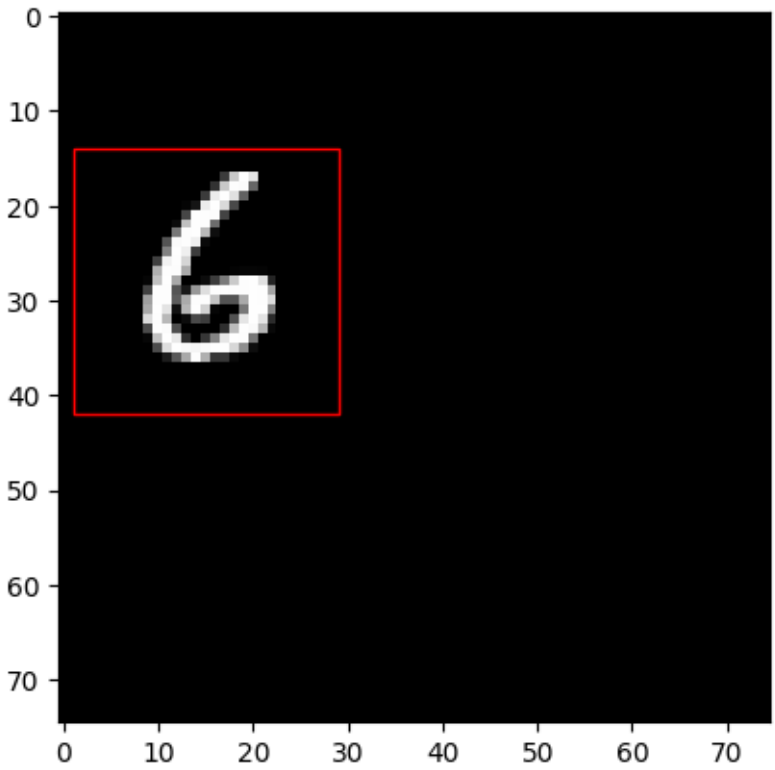
[8]:
```
data = {}                                                    # appending the bbox␣
 ↪cordinates into the data dictionary
for idx, names in enumerate(df["image_path"]):
  if names not in data:
    data["train/train/" + names] = []
    x1 = int(df["x_mark"][idx])
    y1 = int(df["y_mark"][idx])
    x2 = int(df["width"][idx])
    y2 = int(df["height"][idx])
    data["train/train/" + names].append(x1)
    data["train/train/" + names].append(y1)
    data["train/train/" + names].append(x2)
    data["train/train/" + names].append(y2)
```

[9]:
```
images = []                          # appending the images into the images array
for keys in data.keys():
  img_arr = cv.imread(keys, cv.IMREAD_GRAYSCALE)
  images.append(img_arr)
```

[10]:
```
bbox = []
for bboxs in data.keys():
  bbox.append(data[bboxs])
```

[11]:
```
images = np.array(images)
bbox = np.array(bbox)
```

[12]:
```
def test_plot(img_number):
    x1 = bbox[img_number][0]
    y1 = bbox[img_number][1]
    width = bbox[img_number][2]
    height = bbox[img_number][3]
    plt.subplot(1,1,1)
    plt.imshow(images[img_number], cmap='gray')
    plt.gca().add_patch(patches.Rectangle((x1, y1), width-x1, height-y1,␣
 ↪edgecolor='r', facecolor="none", linewidth=1))
plt.show()
test_plot(6000)
```



[13]:
```
images = np.expand_dims(images, axis=3)        # adding a new axis to the images␣
↪array
```

[14]:
```
images = images/255                # normalising the images
```

Splitting and Model Building

[15]:
```
x_train, x_test, y_train, y_test = train_test_split(images, bbox, test_size=0.
↪2, random_state=44)
```

[16]:
```
model = Sequential()

model.add(Conv2D(32, (3,3), input_shape=(75,75,1), activation="relu"))
model.add(MaxPool2D(2,2))
model.add(Conv2D(64, (3,3), activation="relu"))
model.add(MaxPool2D(2,2))
model.add(Conv2D(128, (3,3), activation="relu"))
model.add(MaxPool2D(2,2))
model.add(GlobalAveragePooling2D())
model.add(Dense(256, activation="relu"))
model.add(Dense(4, activation="relu"))
```

[17]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 73, 73, 32)        320

 max_pooling2d (MaxPooling2D  (None, 36, 36, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 34, 34, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 17, 17, 64)       0
 2D)

 conv2d_2 (Conv2D)           (None, 15, 15, 128)       73856

 max_pooling2d_2 (MaxPooling  (None, 7, 7, 128)        0
 2D)

 global_average_pooling2d (G  (None, 128)              0
 lobalAveragePooling2D)

 dense (Dense)               (None, 256)               33024

 dense_1 (Dense)             (None, 4)                 1028

=================================================================
Total params: 126,724
Trainable params: 126,724
Non-trainable params: 0
_________________________________________________________________
```

[18]:
```
class CustomCallbacks(keras.callbacks.Callback):
  def on_epoch_end(self,epoch,logs={}):
    num = np.random.randint(0, 10000)
    bbox = model.predict(x_test[num].reshape(1, 75, 75, 1))
    fig, ax = plt.subplots(1)
    ax.imshow(x_test[num])
    x1 = int(bbox[0][0])
    y1 = int(bbox[0][1])
    x2 = int(bbox[0][2])
    y2 = int(bbox[0][3])
    rect = patches.Rectangle((x1, y1), x2-x1, y2-y1,␣
 ↪linewidth=1,edgecolor='r',facecolor="none")
    ax.add_patch(rect)
```

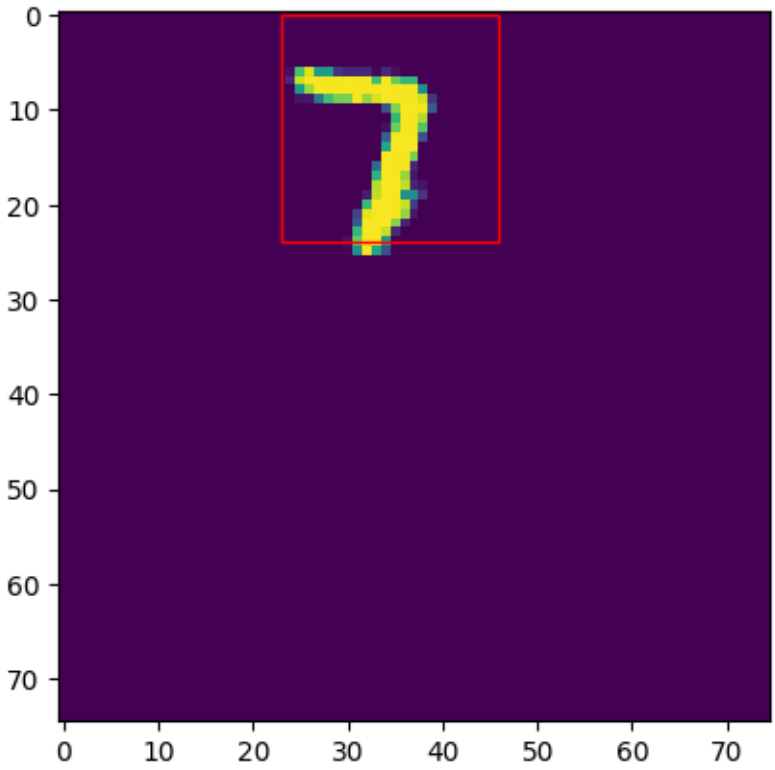
```
    plt.show()
```

[19]:
```
model.compile(optimizer="adam", loss='mae')
```

[20]:
```
history = model.fit(x_train, y_train, validation_data=(x_test, y_test),␣
↪epochs=30, callbacks=[CustomCallbacks()])
```

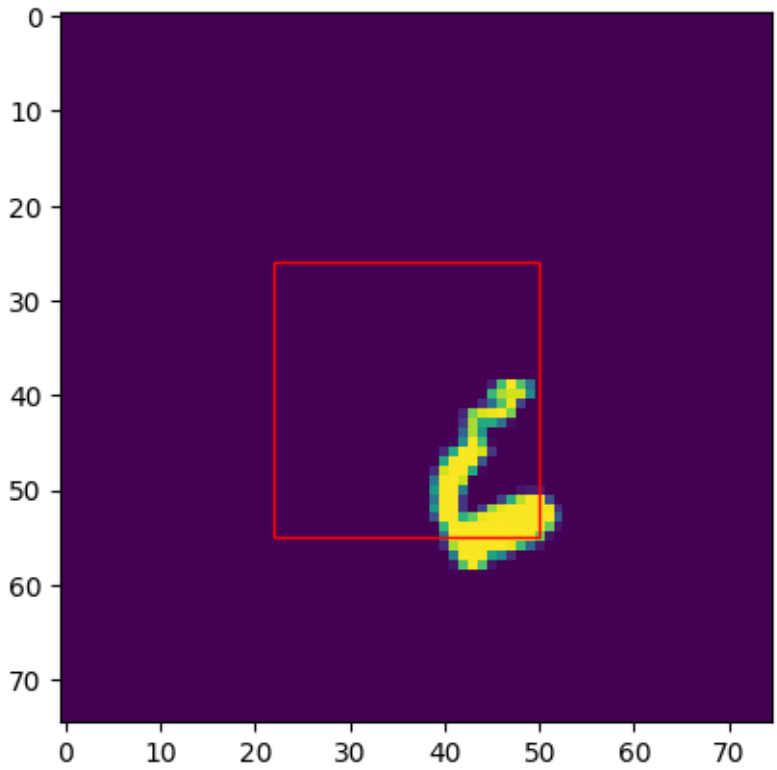
```
Epoch 1/30
1/1 [==============================] - 1s 517ms/step
```

```
<Figure size 1000x500 with 0 Axes>
```



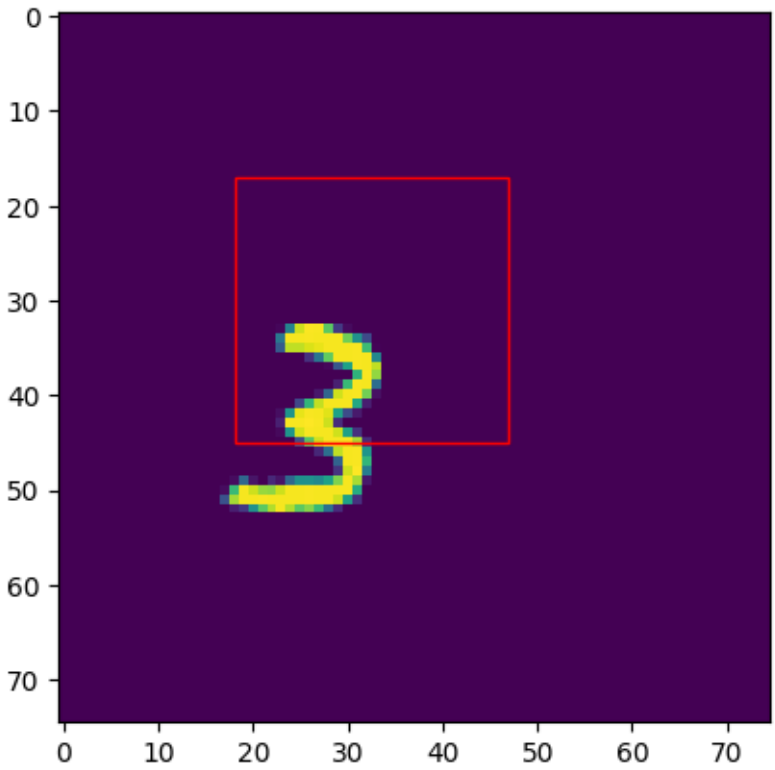

```
1500/1500 [==============================] - 23s 12ms/step - loss: 11.0341 -
val_loss: 6.7351
Epoch 2/30
1/1 [==============================] - 0s 19ms/step
```

```
<Figure size 1000x500 with 0 Axes>
```



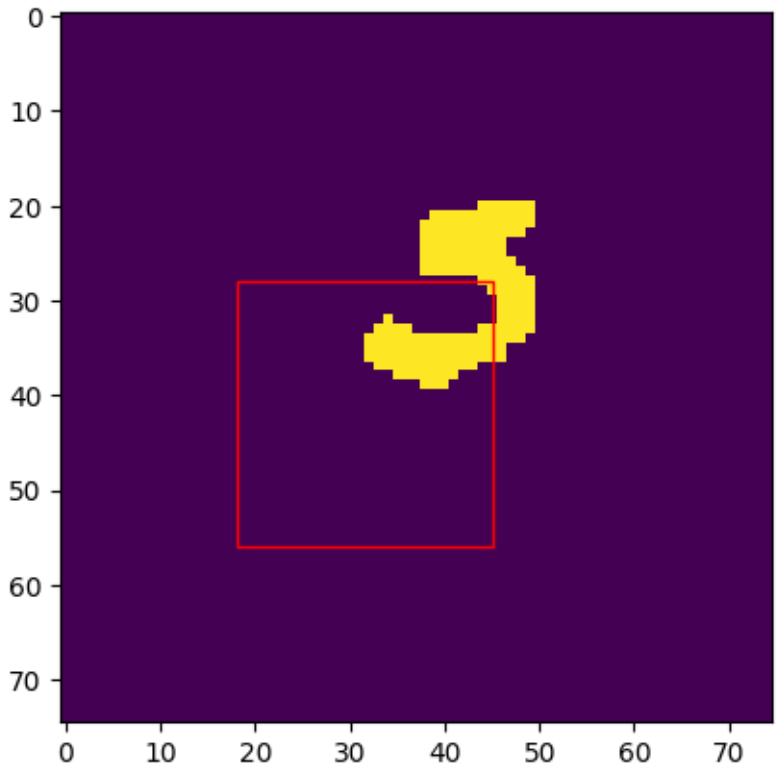

```
1500/1500 [==============================] - 13s 9ms/step - loss: 6.0866 -
val_loss: 5.7306
Epoch 3/30
1/1 [==============================] - 0s 31ms/step

<Figure size 1000x500 with 0 Axes>
```



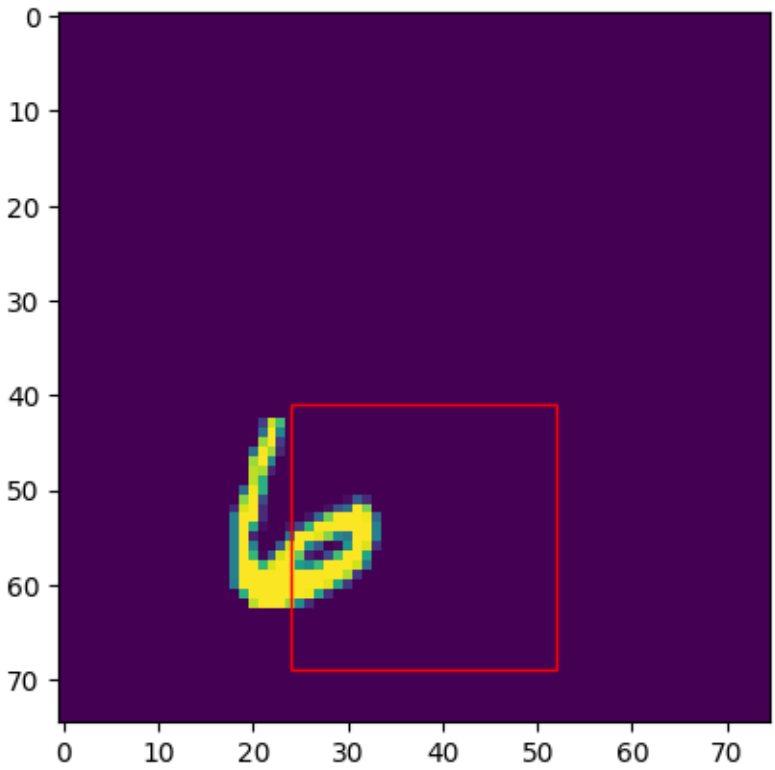

```
1500/1500 [==============================] - 10s 7ms/step - loss: 5.2336 -
val_loss: 5.1354
Epoch 4/30
1/1 [==============================] - 0s 20ms/step
```

```
<Figure size 1000x500 with 0 Axes>
```



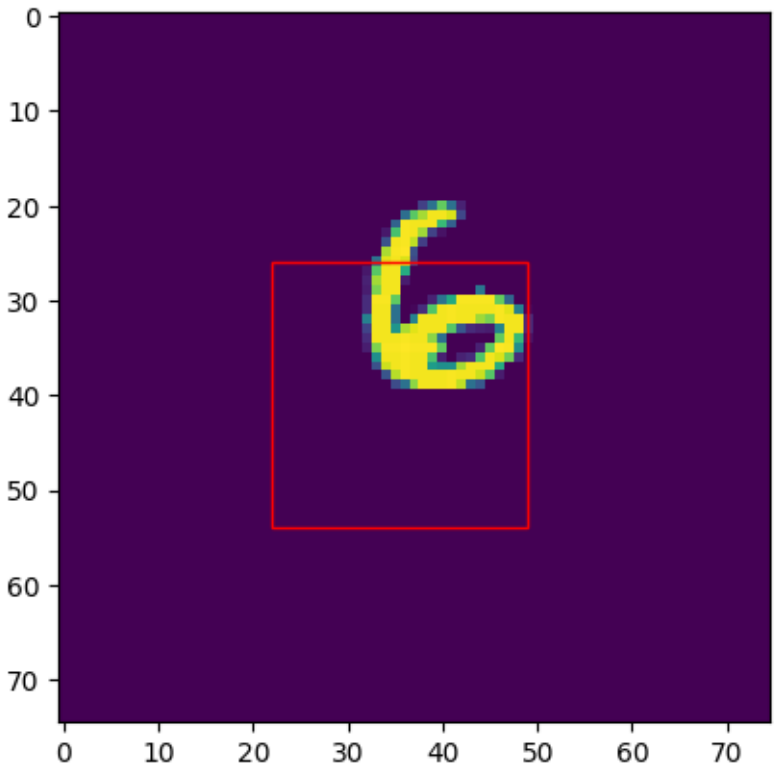

```
1500/1500 [==============================] - 11s 7ms/step - loss: 4.7971 -
val_loss: 5.0530
Epoch 5/30
1/1 [==============================] - 0s 18ms/step
```

```
<Figure size 1000x500 with 0 Axes>
```



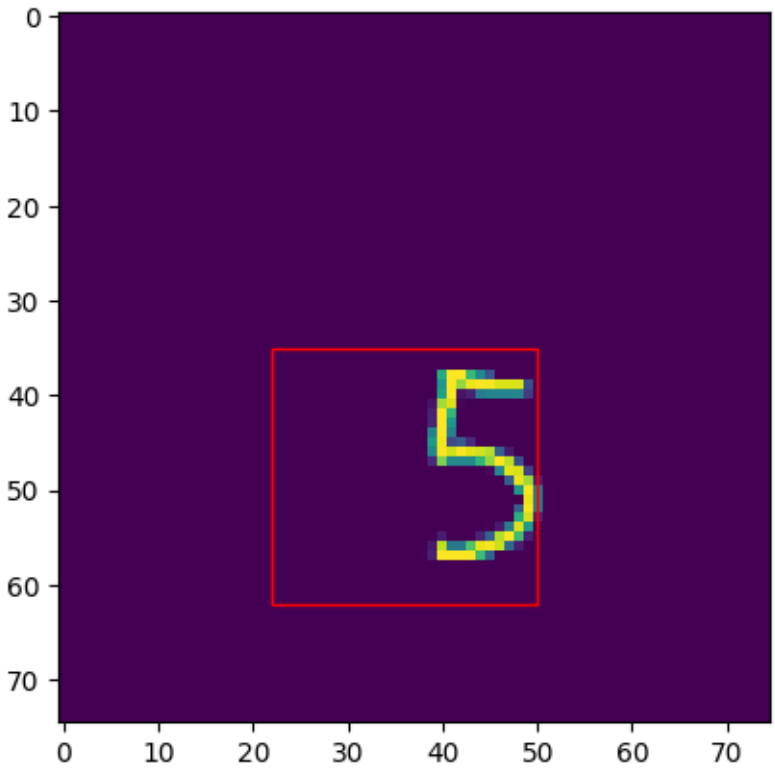

```
1500/1500 [==============================] - 11s 7ms/step - loss: 4.5436 -
val_loss: 4.9210
Epoch 6/30
1/1 [==============================] - 0s 21ms/step
```

```
<Figure size 1000x500 with 0 Axes>
```



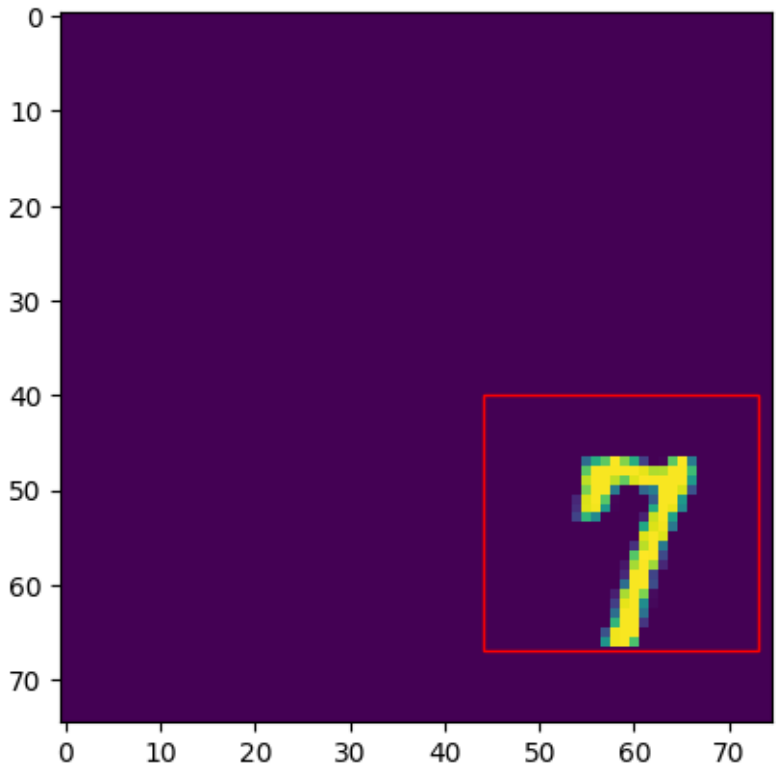

```
1500/1500 [==============================] - 11s 7ms/step - loss: 4.3893 -
val_loss: 4.4408
Epoch 7/30
1/1 [==============================] - 0s 26ms/step
```

```
<Figure size 1000x500 with 0 Axes>
```



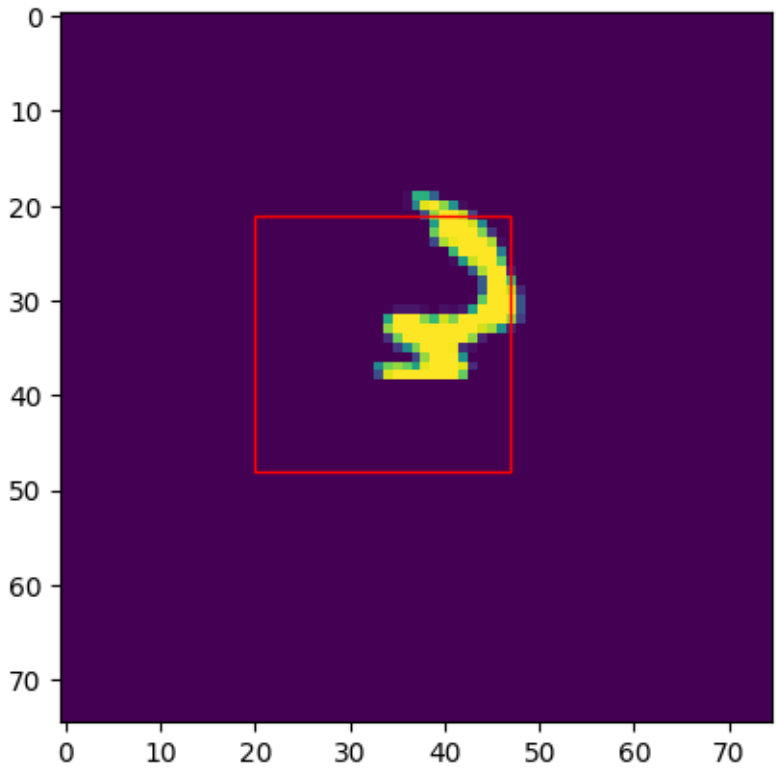

```
1500/1500 [==============================] - 11s 7ms/step - loss: 4.3286 -
val_loss: 4.3952
Epoch 8/30
1/1 [==============================] - 0s 19ms/step
```

```
<Figure size 1000x500 with 0 Axes>
```



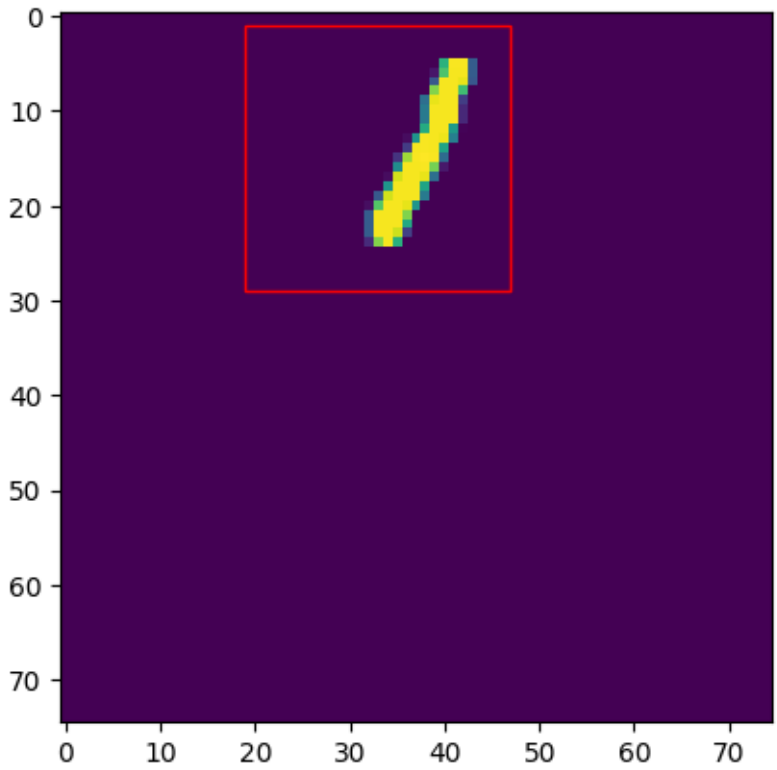

```
1500/1500 [==============================] - 11s 7ms/step - loss: 4.2556 -
val_loss: 4.3557
Epoch 9/30
1/1 [==============================] - 0s 38ms/step
```

```
<Figure size 1000x500 with 0 Axes>
```



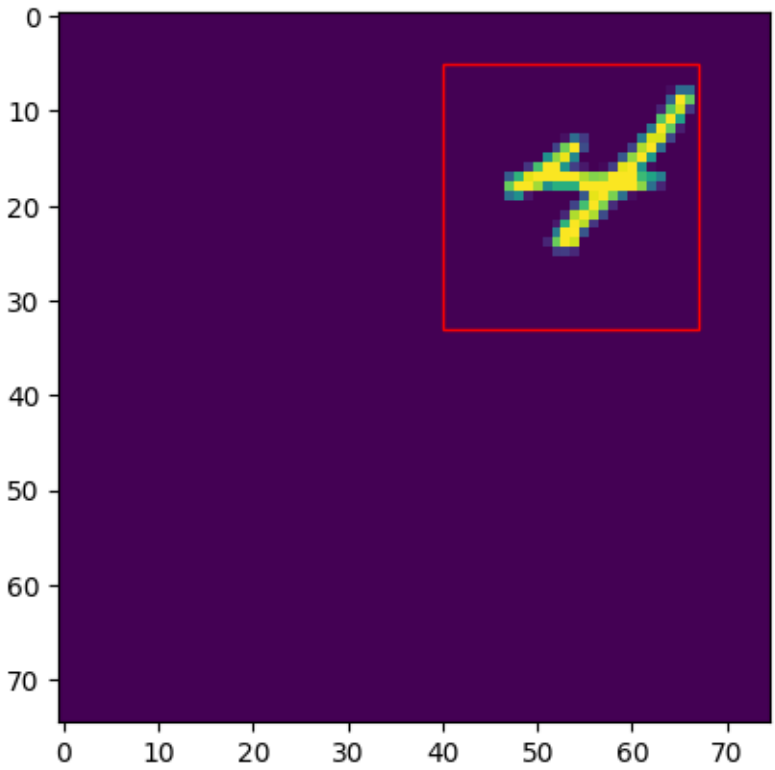

```
1500/1500 [==============================] - 14s 10ms/step - loss: 4.2034 -
val_loss: 4.2513
Epoch 10/30
1/1 [==============================] - 0s 21ms/step

<Figure size 1000x500 with 0 Axes>
```



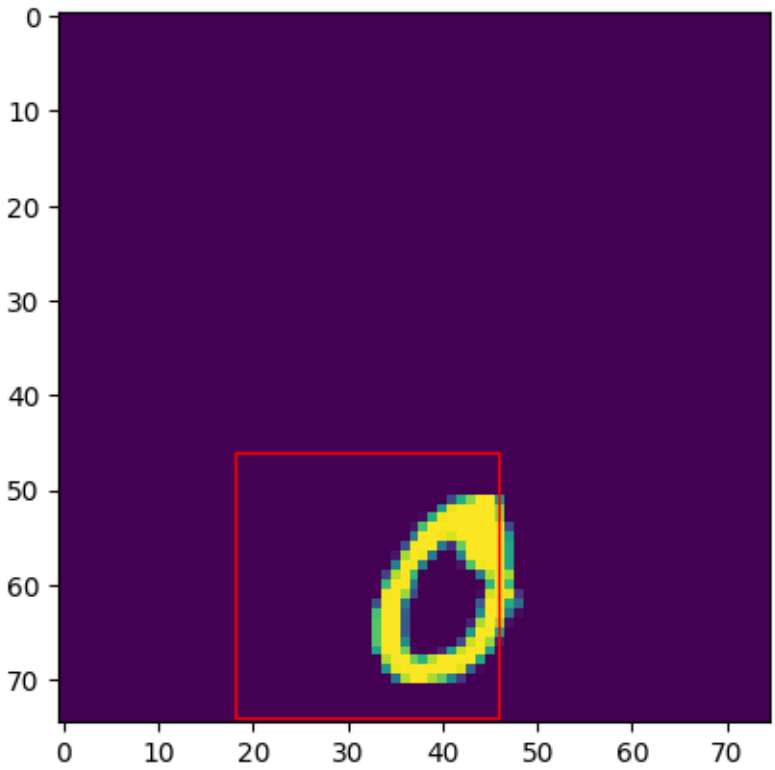

```
1500/1500 [==============================] - 11s 8ms/step - loss: 4.1453 -
val_loss: 4.3206
Epoch 11/30
1/1 [==============================] - 0s 26ms/step

<Figure size 1000x500 with 0 Axes>
```



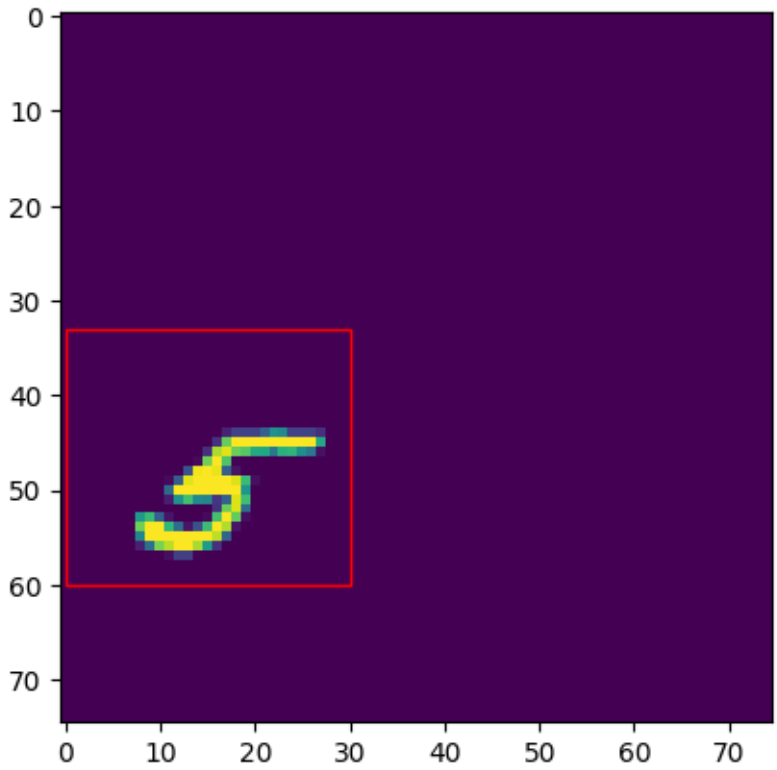

```
1500/1500 [==============================] - 11s 7ms/step - loss: 4.1277 -
val_loss: 4.0900
Epoch 12/30
1/1 [==============================] - 0s 29ms/step
```

```
<Figure size 1000x500 with 0 Axes>
```



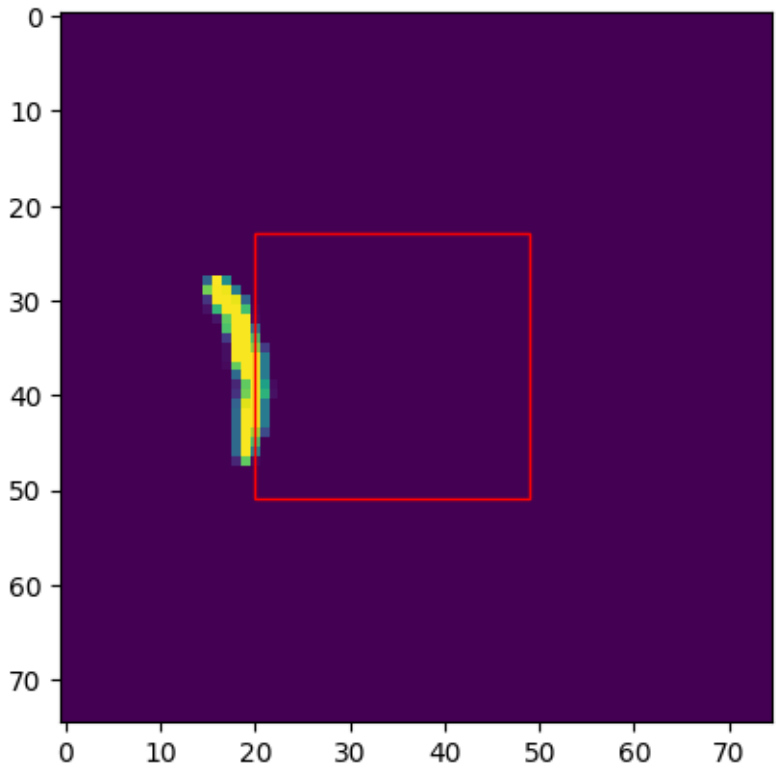

```
1500/1500 [==============================] - 11s 8ms/step - loss: 4.0865 -
val_loss: 4.0703
Epoch 13/30
1/1 [==============================] - 0s 21ms/step

<Figure size 1000x500 with 0 Axes>
```



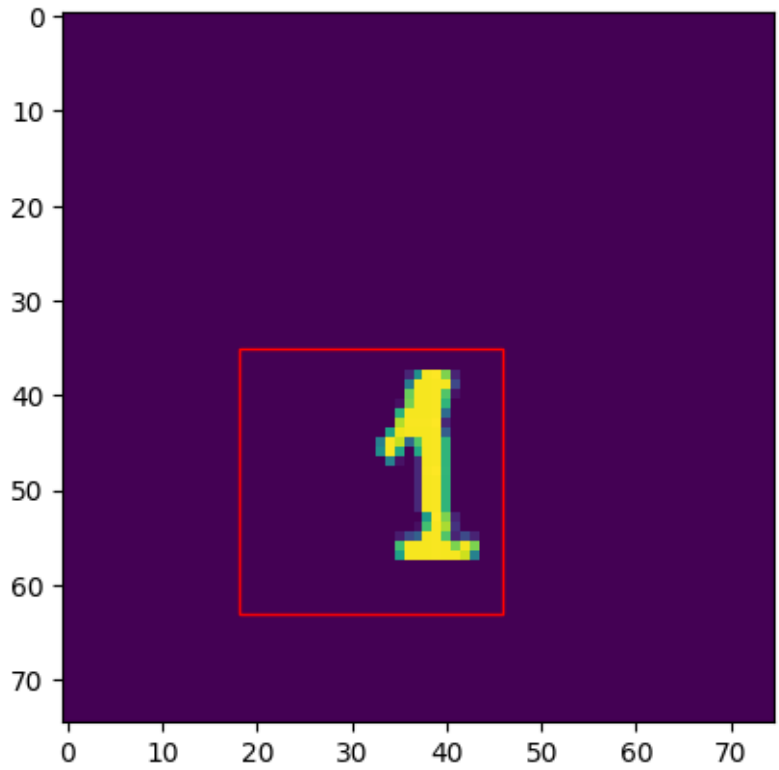

```
1500/1500 [==============================] - 11s 7ms/step - loss: 4.0419 -
val_loss: 4.0991
Epoch 14/30
1/1 [==============================] - 0s 18ms/step
```

```
<Figure size 1000x500 with 0 Axes>
```



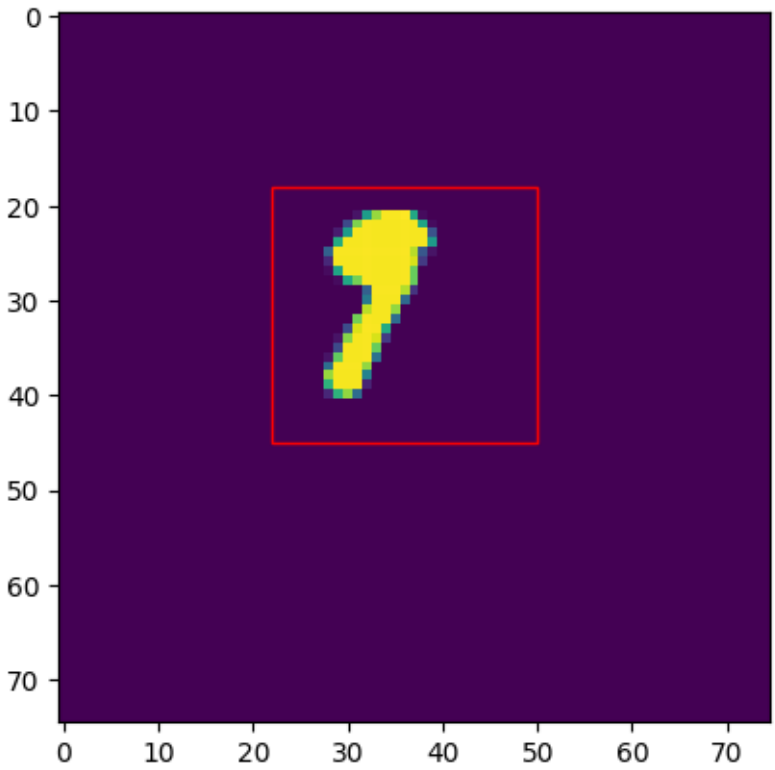

```
1500/1500 [==============================] - 11s 7ms/step - loss: 4.0640 -
val_loss: 3.9957
Epoch 15/30
1/1 [==============================] - 0s 22ms/step
```

```
<Figure size 1000x500 with 0 Axes>
```



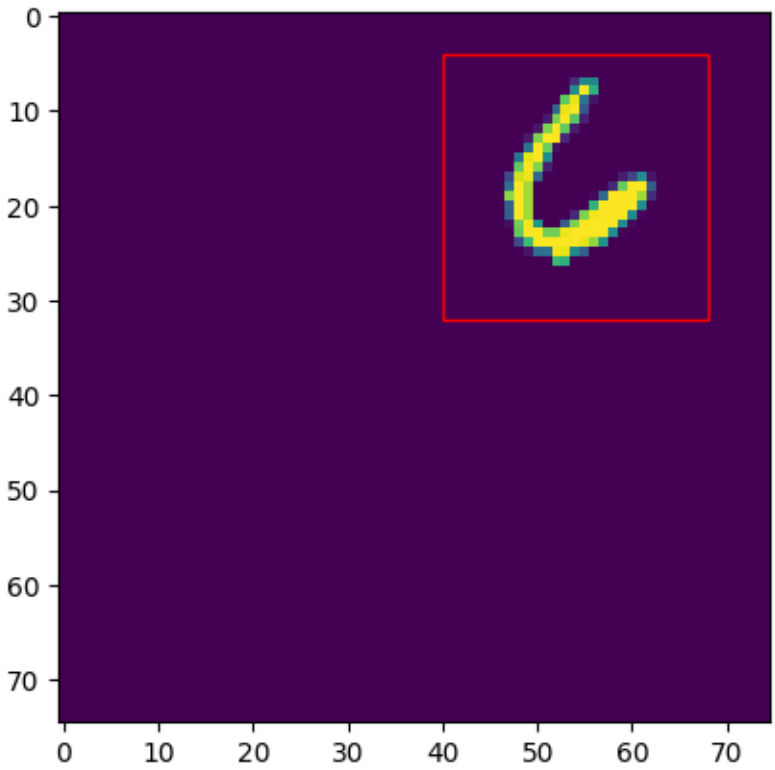

```
1500/1500 [==============================] - 11s 8ms/step - loss: 4.0109 -
val_loss: 3.9401
Epoch 16/30
1/1 [==============================] - 0s 21ms/step

<Figure size 1000x500 with 0 Axes>
```



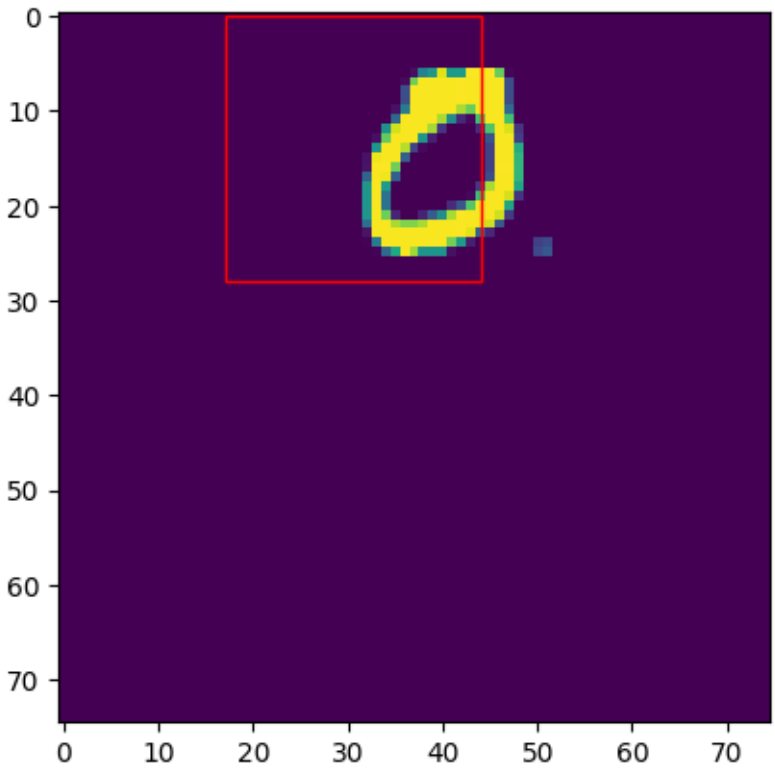

```
1500/1500 [==============================] - 11s 7ms/step - loss: 3.9985 -
val_loss: 4.0390
Epoch 17/30
1/1 [==============================] - 0s 18ms/step

<Figure size 1000x500 with 0 Axes>
```



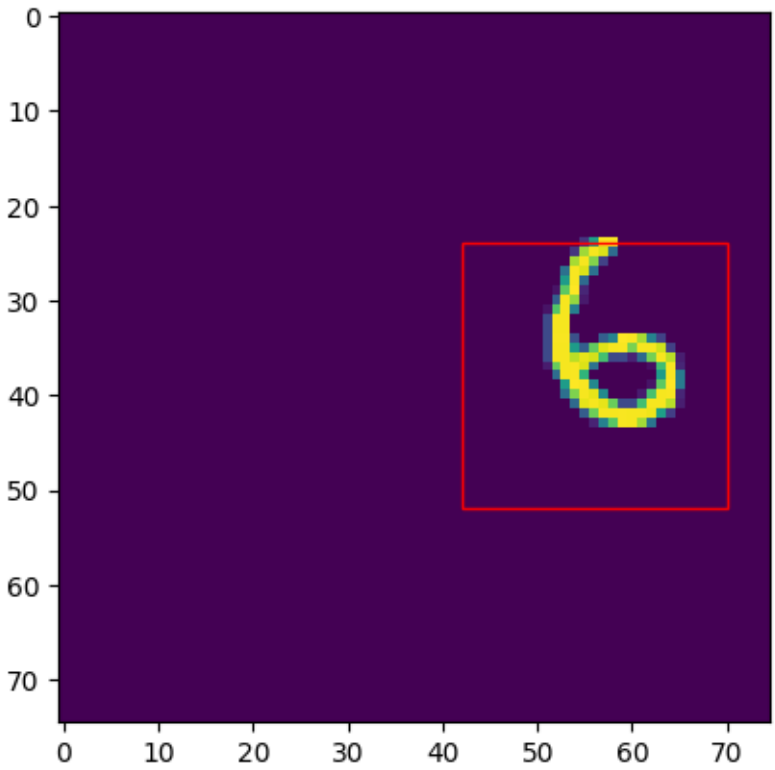

```
1500/1500 [==============================] - 11s 7ms/step - loss: 3.9795 -
val_loss: 3.9221
Epoch 18/30
1/1 [==============================] - 0s 27ms/step
```

```
<Figure size 1000x500 with 0 Axes>
```



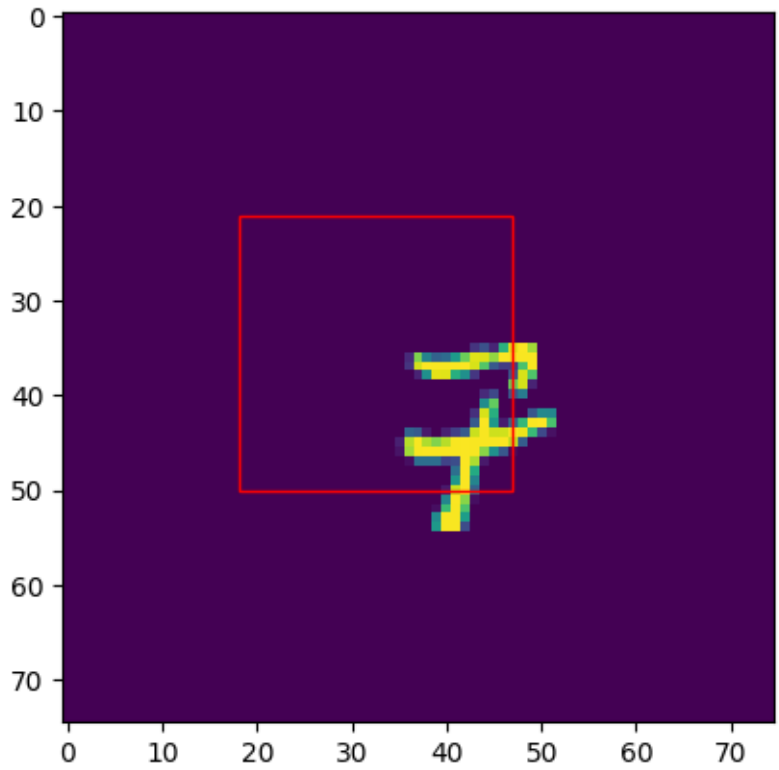

```
1500/1500 [==============================] - 11s 7ms/step - loss: 3.9545 -
val_loss: 3.9476
Epoch 19/30
1/1 [==============================] - 0s 29ms/step
```

```
<Figure size 1000x500 with 0 Axes>
```



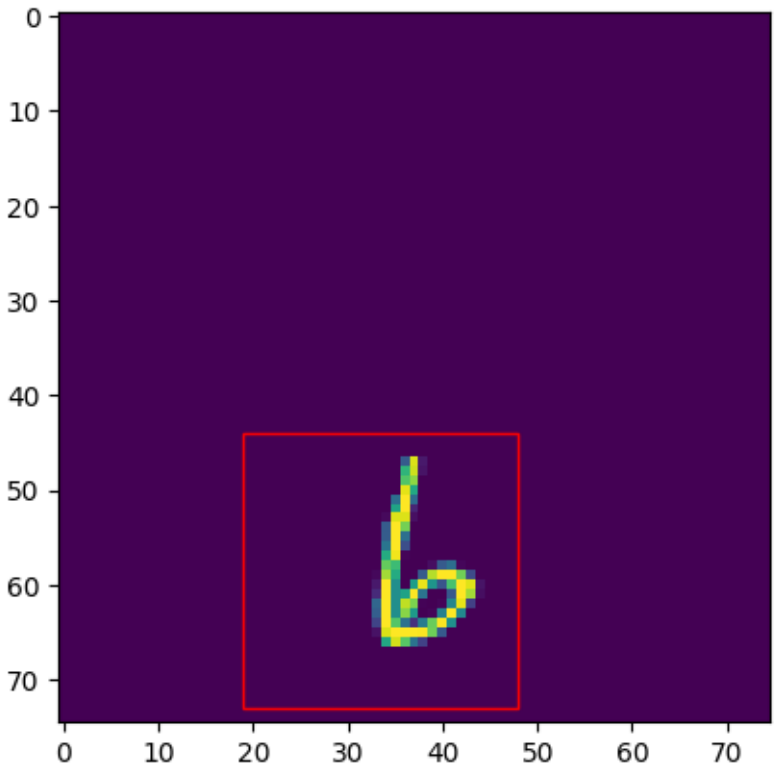

```
1500/1500 [==============================] - 12s 8ms/step - loss: 3.9382 -
val_loss: 3.9146
Epoch 20/30
1/1 [==============================] - 0s 19ms/step
```

```
<Figure size 1000x500 with 0 Axes>
```



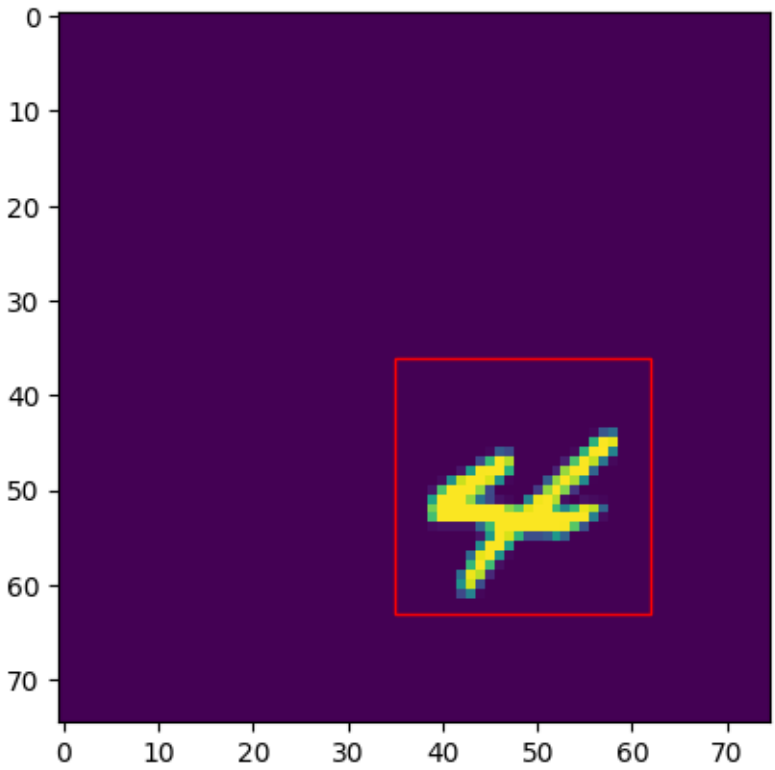

```
1500/1500 [==============================] - 12s 8ms/step - loss: 3.9172 -
val_loss: 3.8593
Epoch 21/30
1/1 [==============================] - 0s 21ms/step
```

```
<Figure size 1000x500 with 0 Axes>
```



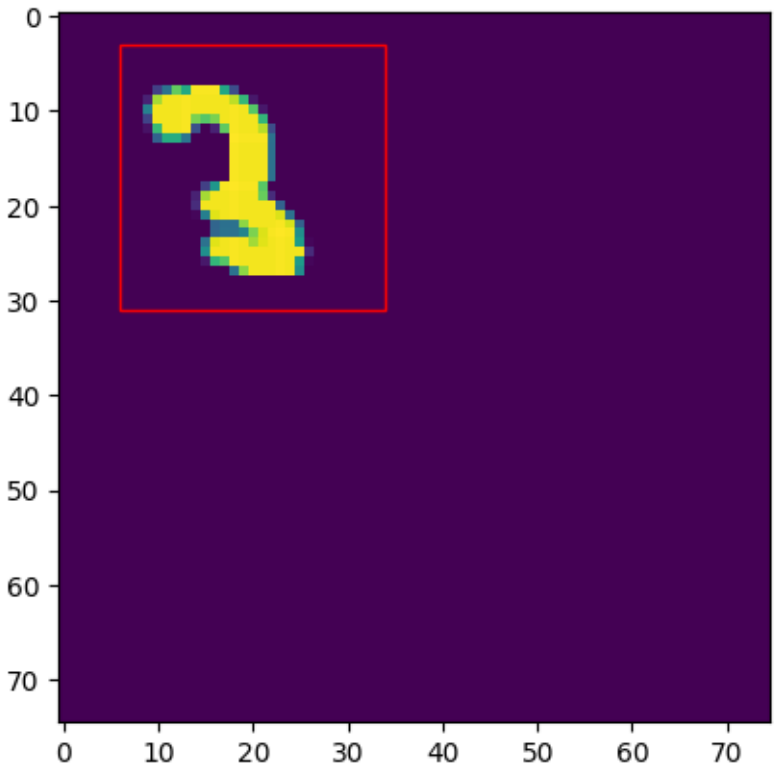

```
1500/1500 [==============================] - 12s 8ms/step - loss: 3.9241 -
val_loss: 3.9327
Epoch 22/30
1/1 [==============================] - 0s 22ms/step

<Figure size 1000x500 with 0 Axes>
```



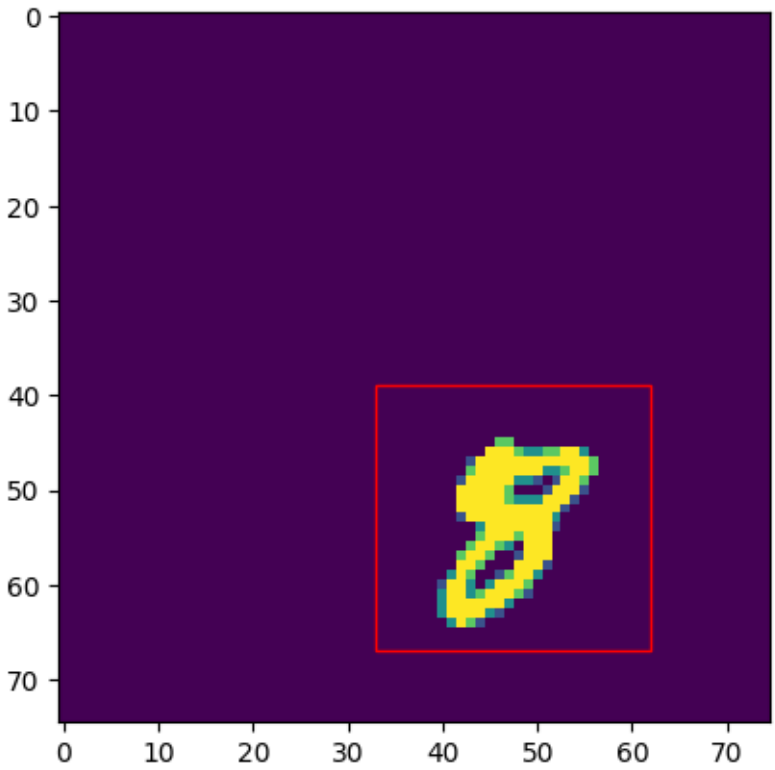

```
1500/1500 [==============================] - 12s 8ms/step - loss: 3.8779 -
val_loss: 3.9742
Epoch 23/30
1/1 [==============================] - 0s 19ms/step
```

```
<Figure size 1000x500 with 0 Axes>
```



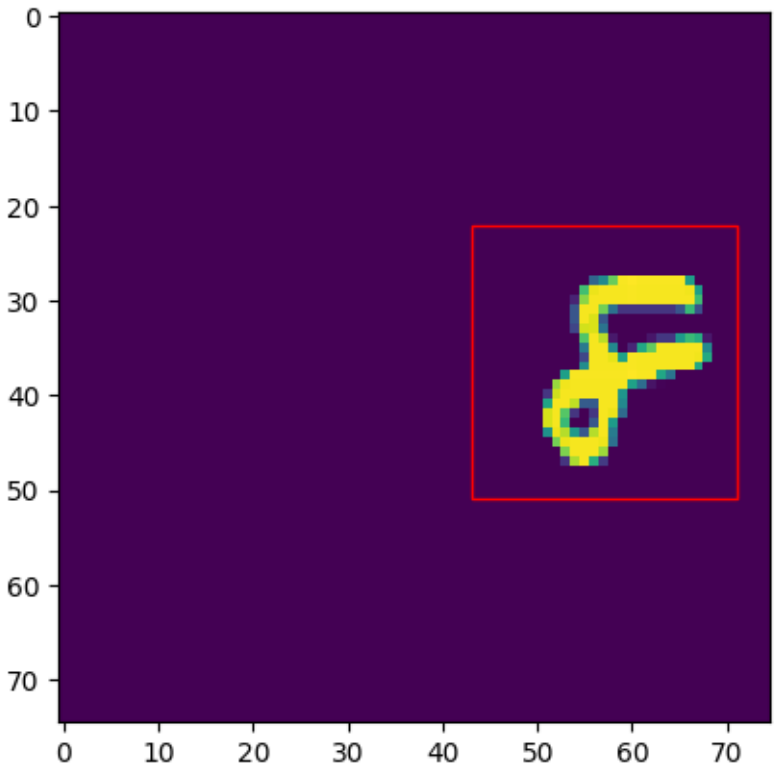

```
1500/1500 [==============================] - 11s 8ms/step - loss: 3.8836 -
val_loss: 3.8842
Epoch 24/30
1/1 [==============================] - 0s 20ms/step
```

```
<Figure size 1000x500 with 0 Axes>
```



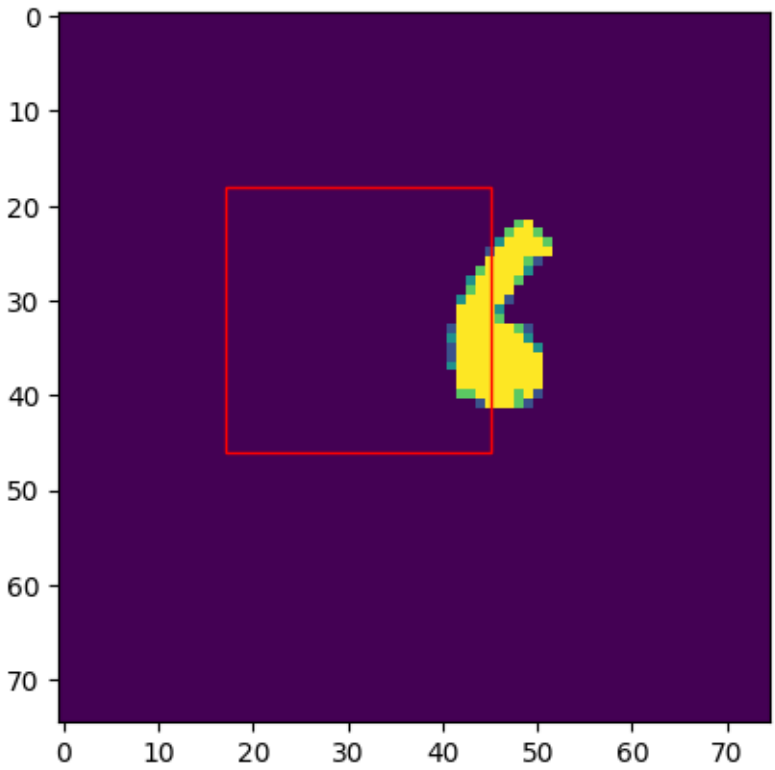

```
1500/1500 [==============================] - 11s 8ms/step - loss: 3.8524 -
val_loss: 3.8593
Epoch 25/30
1/1 [==============================] - 0s 22ms/step

<Figure size 1000x500 with 0 Axes>
```



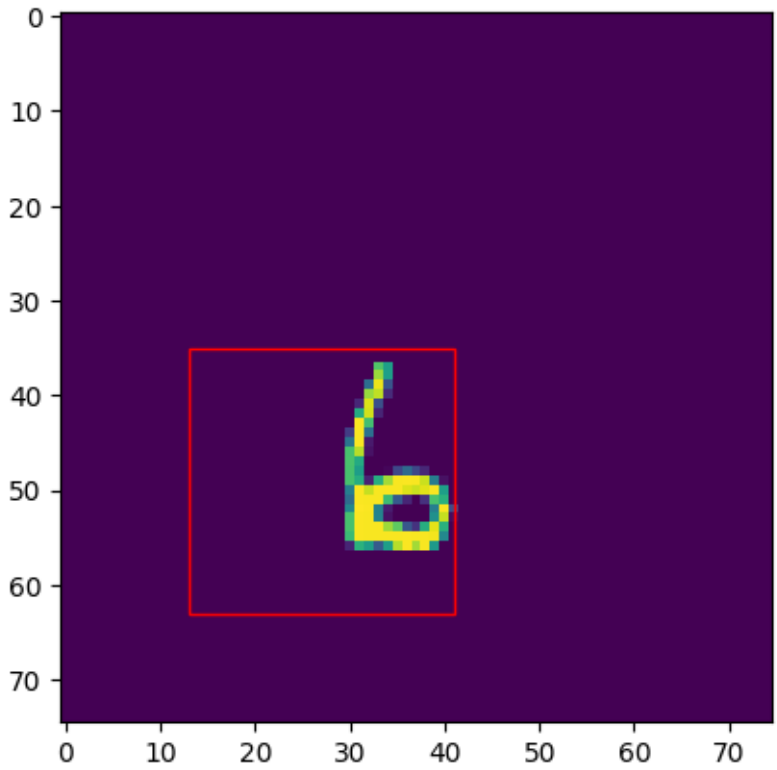

```
1500/1500 [==============================] - 11s 8ms/step - loss: 3.8601 -
val_loss: 3.8985
Epoch 26/30
1/1 [==============================] - 0s 36ms/step
```

```
<Figure size 1000x500 with 0 Axes>
```



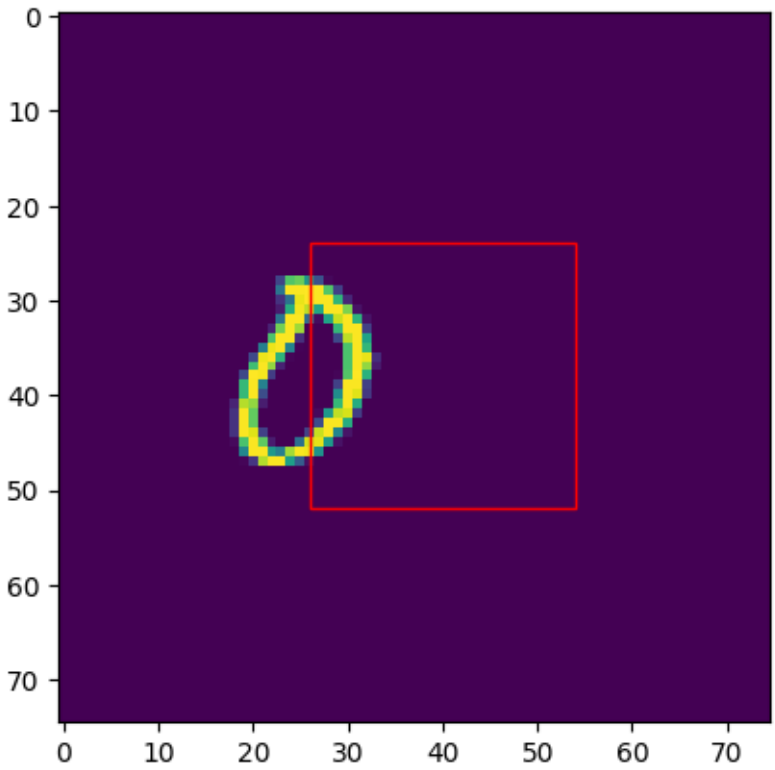

```
1500/1500 [==============================] - 12s 8ms/step - loss: 3.8284 -
val_loss: 3.8247
Epoch 27/30
1/1 [==============================] - 0s 42ms/step

<Figure size 1000x500 with 0 Axes>
```



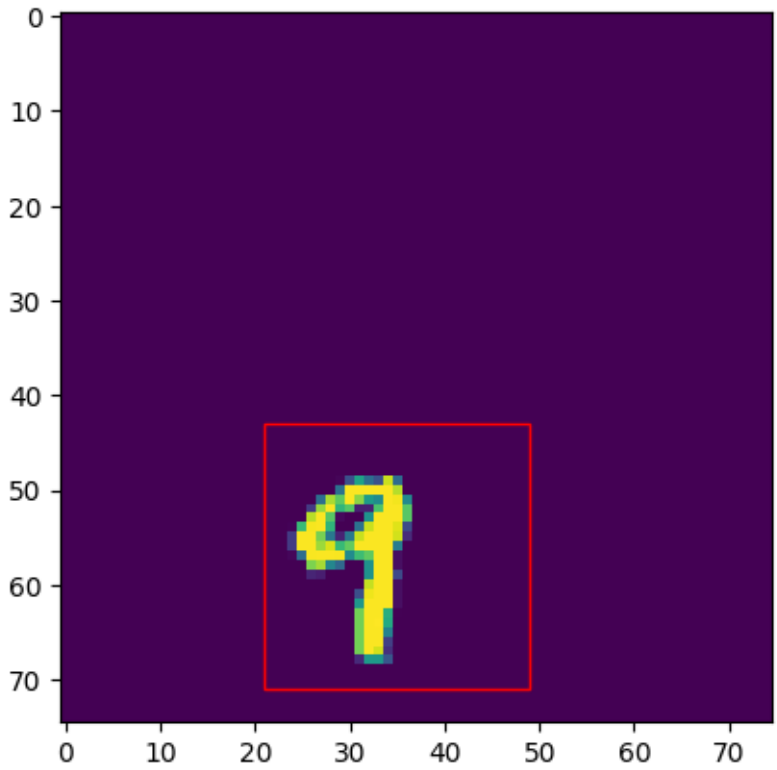

```
1500/1500 [==============================] - 11s 7ms/step - loss: 3.8308 -
val_loss: 3.8820
Epoch 28/30
1/1 [==============================] - 0s 19ms/step

<Figure size 1000x500 with 0 Axes>
```



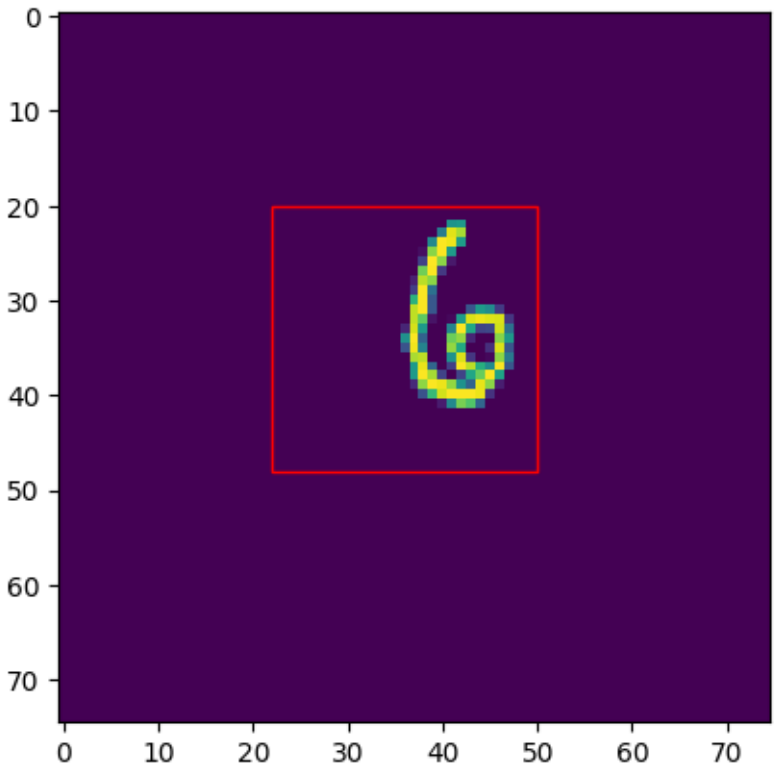

```
1500/1500 [==============================] - 12s 8ms/step - loss: 3.8187 -
val_loss: 4.0059
Epoch 29/30
1/1 [==============================] - 0s 22ms/step
```

```
<Figure size 1000x500 with 0 Axes>
```



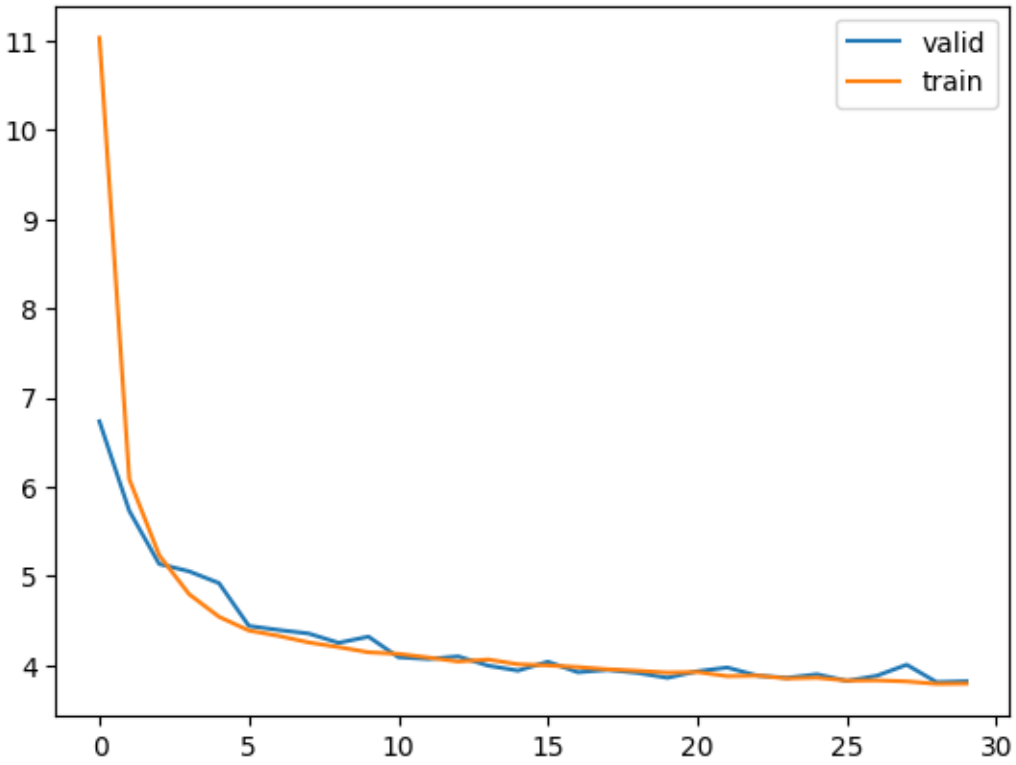

```
1500/1500 [==============================] - 12s 8ms/step - loss: 3.7913 -
val_loss: 3.8108
Epoch 30/30
1/1 [==============================] - 0s 19ms/step

<Figure size 1000x500 with 0 Axes>
```



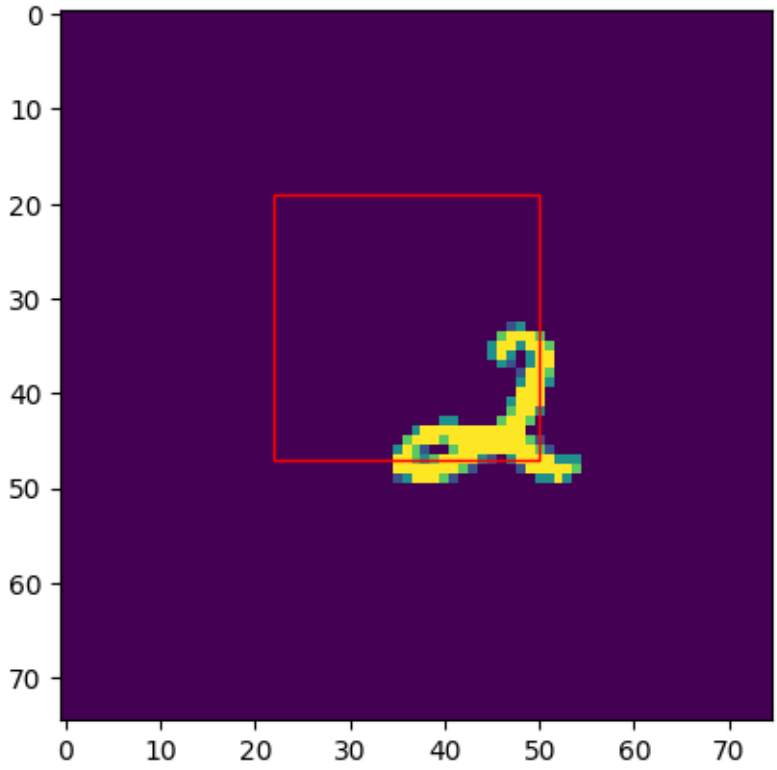

```
1500/1500 [==============================] - 11s 8ms/step - loss: 3.7943 -
val_loss: 3.8207
```

[21]:
```
plt.plot(history.history["val_loss"])
plt.plot(history.history["loss"])
plt.legend(["valid", "train"])
plt.show()
```



[22]:
```
def plot_prediction(img_number):
  prediction = model.predict(x_test[img_number].reshape(1, 75, 75, 1))
  x1 = int(prediction[0][0])
  y1 = int(prediction[0][1])
  x2 = int(prediction[0][2])
  y2 = int(prediction[0][3])
  plt.imshow(x_test[img_number])
  plt.gca().add_patch(patches.Rectangle((x1, y1), x2-x1, y2-y1,␣
 ↪linewidth=1,edgecolor='r',facecolor="none"))
  plt.show()
plot_prediction(30)
```

```
1/1 [==============================] - 0s 19ms/step
```


0
10
20
30
40
50
60
70
0
10
20
30
40
50
60
70