In [1]:
```
import pprint
import rasterio
from rasterio import features
import rasterio.warp
import matplotlib.pyplot as plt
import numpy as np
from scipy.ndimage import zoom
import os

import torch
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
import torchvision
from torchvision import datasets, transforms
from torch.utils.data import Dataset
from torch.optim.lr_scheduler import StepLR
```

In [2]:
```
# Base values
channel_descriptions = ['M11', 'I2', 'I1', 'NDVI_last', 'EVI2_last', 'total precipi

min_values = [np.float32(-100.0),
  np.float32(-100.0),
  np.float32(-100.0),
  np.float32(-9863.268),
  np.float32(-4422.217),
  np.float32(0.0),
  np.float32(0.0),
  np.float32(0.0),
  np.float32(0.0),
  np.float32(0.0),
  np.float32(0.0),
  np.float32(0.0),
  np.float32(0.0),
  np.float32(0.0),
  np.float32(-84.0),
  np.float32(-6.72),
  np.float32(1.0),
  np.float32(0.0),
  np.float32(0.0),
  np.float32(-89.999214),
  np.float32(-13.984883),
  np.float32(0.0),
  np.float32(0.0)]

max_values = [np.float32(15976.0),
  np.float32(15799.0),
  np.float32(15744.0),
  np.float32(9975.073),
  np.float32(9856.787),
  np.float32(122.0),
  np.float32(16.2),
  np.float32(360.0),
  np.float32(311.8),
```

```
  np.float32(325.4),
  np.float32(122.0),
  np.float32(0.01888),
  np.float32(63.85685),
  np.float32(359.42383),
  np.float32(4268.336),
  np.float32(8.28),
  np.float32(17.0),
  np.float32(204.1875),
  np.float32(14.295916),
  np.float32(89.98897),
  np.float32(39.505894),
  np.float32(0.0122514665),
  np.float32(2218.0)]
```

In [3]:

```
fire_folders = []
look_back = 5   # 5 days sequence
all_frames = []
data_limit, loop_counter, loop_start = 7, 0, 0
channel_descriptions = None
base_path = "./data"

target_shape_h, target_shape_w = 128, 128

print('Loading...')

for fire_folder in os.listdir(base_path):
    loop_counter += 1
    if loop_counter - loop_start > data_limit:
        break
    if loop_counter < loop_start:
        continue
    print('Progress: {0}/{1} ({2})'.format(loop_counter - loop_start, data_limit, l
    fire_folders.append(fire_folder)
    for image_name in os.listdir(base_path + f"/{fire_folder}"):
        file_path = base_path + f"/{fire_folder}/{image_name}"

        with rasterio.open(file_path, 'r') as geotiff:
            src = geotiff.read()
            channel_descriptions = geotiff.descriptions
            zoom_factor = (1, target_shape_h / src.shape[1], target_shape_w / src.s
            resized_src = zoom(src, zoom_factor, order=1)
            resized_src = np.nan_to_num(resized_src, copy=True)
            all_frames.append(resized_src)

print(f'Loading done! Count = {len(all_frames)} | Shape = {all_frames[0].shape}')

data_frames = np.stack(all_frames)
print(data_frames.shape)
```

```
Loading...
Progress: 1/7 (1)
Progress: 2/7 (2)
Progress: 3/7 (3)
Progress: 4/7 (4)
Progress: 5/7 (5)
Progress: 6/7 (6)
Progress: 7/7 (7)
Loading done! Count = 168 | Shape = (23, 128, 128)
(168, 23, 128, 128)
```

In [4]:

```
from sklearn.preprocessing import MinMaxScaler, minmax_scale

#data_frames = (data_frames - min_bound) / (max_bound - min_bound)

for c in range(23):
    data_frames[:, c, :, :] = (data_frames[:, c, :, :] - min_values[c]) / (max_valu
```

In [5]:

```
np.min(data_frames), np.max(data_frames)
```

Out[5]:

```
(np.float32(0.0), np.float32(1.0))
```

In [6]:

```
X = []
Y = []

for t in range(0, data_frames.shape[0] - look_back):
    x_seq = data_frames[t:t+look_back]               # shape: (5, 23, H, W)
    y_mask = data_frames[t + look_back, 22]          # fire mask from channel 22

    X.append(x_seq)
    Y.append(y_mask)     # binarize

X = np.stack(X)  # shape: (273, 5, 23, 128, 128)
Y = np.expand_dims(np.stack(Y), axis=1)  # shape: (273, 1, 128, 128)

print(X.shape)
print(Y.shape)
```

```
(163, 5, 23, 128, 128)
(163, 1, 128, 128)
```

In [7]:

```
split_index = int(X.shape[0] * 0.8)

X_train = X[:split_index]
X_val = X[split_index:]

Y_train = Y[:split_index]
Y_val = Y[split_index:]

X_train.shape, Y_train.shape, X_val.shape, Y_val.shape
```

Out[7]:

```
((130, 5, 23, 128, 128),
 (130, 1, 128, 128),
 (33, 5, 23, 128, 128),
 (33, 1, 128, 128))
```

In [8]:
```
X_train = torch.tensor(X_train).float()
Y_train = torch.tensor(Y_train).float()

X_val = torch.tensor(X_val).float()
Y_val = torch.tensor(Y_val).float()

X_train.shape, Y_train.shape, X_val.shape, Y_val.shape
```

Out[8]:
```
(torch.Size([130, 5, 23, 128, 128]),
 torch.Size([130, 1, 128, 128]),
 torch.Size([33, 5, 23, 128, 128]),
 torch.Size([33, 1, 128, 128]))
```

In [9]:
```
class WildfireDataset(Dataset):
    def __init__(self, X, Y):
        self.X = X
        self.Y = Y

    def __len__(self):
        return len(self.X)

    def __getitem__(self, i):
        return self.X[i], self.Y[i]

train_dataset = WildfireDataset(X_train, Y_train)
val_dataset = WildfireDataset(X_val, Y_val)

len(train_dataset), len(val_dataset)
```

Out[9]:
```
(130, 33)
```

In [10]:
```
from torch.utils.data import DataLoader

batch_size = 8

train_loader = DataLoader(train_dataset, batch_size=batch_size, shuffle=False)
val_loader = DataLoader(val_dataset, batch_size=batch_size, shuffle=False)
```

In [11]:
```
if torch.cuda.is_available():
    device = torch.device("cuda")
elif torch.mps.is_available():
    device = torch.device("mps")
else:
    device = torch.device("cpu")

device
```

Out[11]:
```
device(type='cuda')
```

In [12]:
```
import convlstm

class Net(nn.Module):
    def __init__(self):
        super().__init__()
        orig_size = (330, 257)
```

```
        self.clstm = convlstm.ConvLSTM(
            input_size=(128, 128),
            input_dim=23,
            hidden_dim=[64],
            kernel_size=(3, 3),
            num_layers=1
        )
        # (8, 64, 128, 128)

        self.head = nn.Sequential(
            nn.Conv2d(64, 1, kernel_size=3, padding=1),
            nn.Sigmoid()
        )
        # (8, 1, 128, 128)

    def forward(self, x):
        batch_size = x.size(0)
        outputs, last_states = self.clstm(x)

        x = outputs[0][:, -1, :, :, :]
        x = self.head(x)
        return x

model = Net().to(device)
```

In [13]:

```
class DiceLoss(nn.Module):
    def __init__(self, smooth=1.0):
        super().__init__()
        self.smooth = smooth
        self.bce = nn.BCELoss()

    def forward(self, y_pred, y_true):
        bce_loss = self.bce(y_pred, y_true)
        # (8, 1, 128, 128)

        y_pred_flat = y_pred.view(y_pred.size(0), -1) # (8, 16384)
        y_true_flat = y_true.view(y_true.size(0), -1) # (8, 16384)

        intersection = (y_pred_flat * y_true_flat).sum(dim=1)
        union = y_pred_flat.sum(dim=1) + y_true_flat.sum(dim=1)

        dice_score = (2 * intersection + self.smooth) / (union + self.smooth)
        dice_loss = 1 - dice_score.mean()

        return dice_loss

class ManualBCELoss(nn.Module):
    def __init__(self, eps=1e-7):
        super(ManualBCELoss, self).__init__()
        self.eps = eps  # to avoid log(0)
        self.weight = 60.0

    def forward(self, y_pred, y_true):
        """
        y_pred: probabilities after sigmoid, shape (B, 1, H, W)
```

```
        y_true: binary targets, shape (B, 1, H, W)
        """
        # Clamp predictions to avoid log(0)
        y_pred = torch.clamp(y_pred, self.eps, 1.0 - self.eps)

        # BCE loss calculation
        loss = - (y_true * self.weight * torch.log(y_pred) + (1 - y_true) * torch.l

        return loss.mean()  # return scalar loss
```

In [14]:
```
lr = 0.001 # 0.0001
num_epochs = 25

loss_fn = ManualBCELoss()

optimizer = torch.optim.Adam(model.parameters(), lr=lr)
print_batch_every = 4

def train(epoch):
    model.train(True)
    print(f"Epoch: {epoch + 1}")
    running_loss = 0.0

    for batch_index, batch in enumerate(train_loader):
        x_batch, y_batch = batch[0].to(device), batch[1].to(device)
        y_pred = model(x_batch)

        loss = loss_fn(y_pred, y_batch)
        running_loss += loss.item()

        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        if batch_index % print_batch_every == (print_batch_every - 1):
            avg_loss_across_batches = running_loss / print_batch_every
            print('Batch {0}, Loss: {1:.3f}'.format(batch_index + 1, avg_loss_acros
            running_loss = 0.0

    print()

def validate():
    model.train(False)
    running_loss = 0.0

    for batch_index, batch in enumerate(val_loader):
        x_batch, y_batch = batch[0].to(device), batch[1].to(device)

        with torch.no_grad():
            y_pred = model(x_batch)
            loss = loss_fn(y_pred, y_batch)
            running_loss += loss.item()

    avg_loss_across_batches = running_loss / len(val_loader)
    print('Val Loss: {0:.3f}'.format(avg_loss_across_batches))
    print('***********************************************************')
```

```
    print()


for epoch in range(num_epochs):
    train(epoch)
    validate()
```

```
Epoch: 1
```

```
C:\Users\neelr\AppData\Local\Temp\ipykernel_38172\689428613.py:19: UserWarning: Conv
erting a tensor with requires_grad=True to a scalar may lead to unexpected behavior.
Consider using tensor.detach() first. (Triggered internally at C:\actions-runner\_wo
rk\pytorch\pytorch\pytorch\aten\src\ATen\native\Scalar.cpp:23.)
  running_loss += loss.item()
```

```
Batch 4, Loss: 0.549
Batch 8, Loss: 0.142
Batch 12, Loss: 0.281
Batch 16, Loss: 0.084

Val Loss: 0.041
**********************************************************

Epoch: 2
Batch 4, Loss: 0.073
Batch 8, Loss: 0.125
Batch 12, Loss: 0.244
Batch 16, Loss: 0.089

Val Loss: 0.057
**********************************************************

Epoch: 3
Batch 4, Loss: 0.080
Batch 8, Loss: 0.120
Batch 12, Loss: 0.233
Batch 16, Loss: 0.086

Val Loss: 0.051
**********************************************************

Epoch: 4
Batch 4, Loss: 0.076
Batch 8, Loss: 0.120
Batch 12, Loss: 0.239
Batch 16, Loss: 0.085

Val Loss: 0.051
**********************************************************

Epoch: 5
Batch 4, Loss: 0.076
Batch 8, Loss: 0.120
Batch 12, Loss: 0.236
Batch 16, Loss: 0.086

Val Loss: 0.052
**********************************************************

Epoch: 6
Batch 4, Loss: 0.076
Batch 8, Loss: 0.120
Batch 12, Loss: 0.236
Batch 16, Loss: 0.086

Val Loss: 0.051
**********************************************************

Epoch: 7
Batch 4, Loss: 0.076
Batch 8, Loss: 0.120
```

```
Batch 12, Loss: 0.236
Batch 16, Loss: 0.086

Val Loss: 0.052
**********************************************************

Epoch: 8
Batch 4, Loss: 0.076
Batch 8, Loss: 0.119
Batch 12, Loss: 0.236
Batch 16, Loss: 0.086

Val Loss: 0.052
**********************************************************

Epoch: 9
Batch 4, Loss: 0.076
Batch 8, Loss: 0.119
Batch 12, Loss: 0.236
Batch 16, Loss: 0.085

Val Loss: 0.052
**********************************************************

Epoch: 10
Batch 4, Loss: 0.076
Batch 8, Loss: 0.119
Batch 12, Loss: 0.235
Batch 16, Loss: 0.085

Val Loss: 0.052
**********************************************************

Epoch: 11
Batch 4, Loss: 0.075
Batch 8, Loss: 0.118
Batch 12, Loss: 0.235
Batch 16, Loss: 0.085

Val Loss: 0.052
**********************************************************

Epoch: 12
Batch 4, Loss: 0.075
Batch 8, Loss: 0.117
Batch 12, Loss: 0.234
Batch 16, Loss: 0.085

Val Loss: 0.052
**********************************************************

Epoch: 13
Batch 4, Loss: 0.075
Batch 8, Loss: 0.116
Batch 12, Loss: 0.234
Batch 16, Loss: 0.085
```

```
Val Loss: 0.052
**********************************************************

Epoch: 14
Batch 4, Loss: 0.074
Batch 8, Loss: 0.115
Batch 12, Loss: 0.232
Batch 16, Loss: 0.084

Val Loss: 0.052
**********************************************************

Epoch: 15
Batch 4, Loss: 0.073
Batch 8, Loss: 0.113
Batch 12, Loss: 0.231
Batch 16, Loss: 0.084

Val Loss: 0.052
**********************************************************

Epoch: 16
Batch 4, Loss: 0.072
Batch 8, Loss: 0.111
Batch 12, Loss: 0.229
Batch 16, Loss: 0.083

Val Loss: 0.052
**********************************************************

Epoch: 17
Batch 4, Loss: 0.071
Batch 8, Loss: 0.109
Batch 12, Loss: 0.226
Batch 16, Loss: 0.081

Val Loss: 0.051
**********************************************************

Epoch: 18
Batch 4, Loss: 0.070
Batch 8, Loss: 0.106
Batch 12, Loss: 0.223
Batch 16, Loss: 0.080

Val Loss: 0.050
**********************************************************

Epoch: 19
Batch 4, Loss: 0.069
Batch 8, Loss: 0.102
Batch 12, Loss: 0.219
Batch 16, Loss: 0.078

Val Loss: 0.049
```

```
**********************************************************

Epoch: 20
Batch 4, Loss: 0.067
Batch 8, Loss: 0.099
Batch 12, Loss: 0.214
Batch 16, Loss: 0.075

Val Loss: 0.047
**********************************************************

Epoch: 21
Batch 4, Loss: 0.066
Batch 8, Loss: 0.095
Batch 12, Loss: 0.208
Batch 16, Loss: 0.072

Val Loss: 0.045
**********************************************************

Epoch: 22
Batch 4, Loss: 0.064
Batch 8, Loss: 0.092
Batch 12, Loss: 0.203
Batch 16, Loss: 0.070

Val Loss: 0.043
**********************************************************

Epoch: 23
Batch 4, Loss: 0.061
Batch 8, Loss: 0.089
Batch 12, Loss: 0.197
Batch 16, Loss: 0.067

Val Loss: 0.040
**********************************************************

Epoch: 24
Batch 4, Loss: 0.059
Batch 8, Loss: 0.085
Batch 12, Loss: 0.190
Batch 16, Loss: 0.063

Val Loss: 0.036
**********************************************************

Epoch: 25
Batch 4, Loss: 0.053
Batch 8, Loss: 0.080
Batch 12, Loss: 0.180
Batch 16, Loss: 0.058

Val Loss: 0.033
**********************************************************
```

In [15]:
```
for batch_index, batch in enumerate(train_loader):
    if batch_index == 4:
        x_batch, y_batch = batch[0].to(device), batch[1].to(device)
        y_pred = model(x_batch)
        print("found")
        break
```

```
found
```

In [16]:
```
torch.max(y_batch), torch.max(y_pred), torch.min(y_batch), torch.min(y_pred)
```

Out[16]:
```
(tensor(0.9261, device='cuda:0'),
 tensor(0.5746, device='cuda:0', grad_fn=<MaxBackward1>),
 tensor(0., device='cuda:0'),
 tensor(0.0054, device='cuda:0', grad_fn=<MinBackward1>))
```

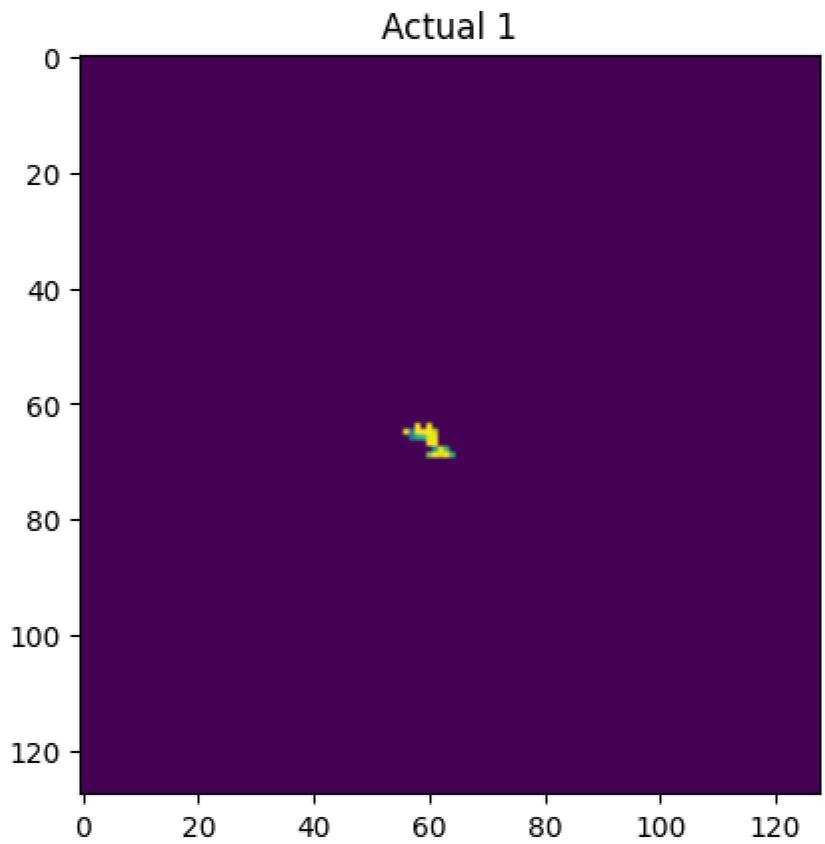
In [17]:
```
y_batch_np = y_batch.cpu().detach().numpy()
y_pred_np = y_pred.cpu().detach().numpy()

y_pred_avg = (np.min(y_pred_np) + np.max(y_pred_np)) / 2.0
#y_pred_np = (y_pred_np - 0.0) / (0.65 - 0.0)

y_batch_np.shape, y_pred_np.shape
```

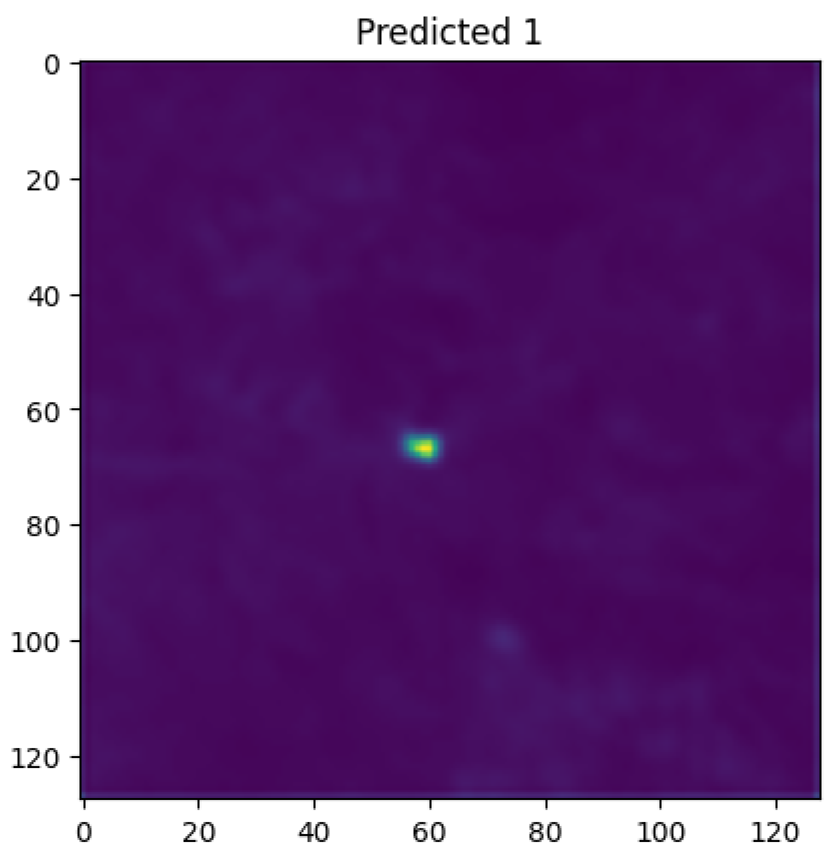
Out[17]:
```
((8, 1, 128, 128), (8, 1, 128, 128))
```

In [18]:
```
np.max(y_pred_np), np.min(y_pred_np)
```

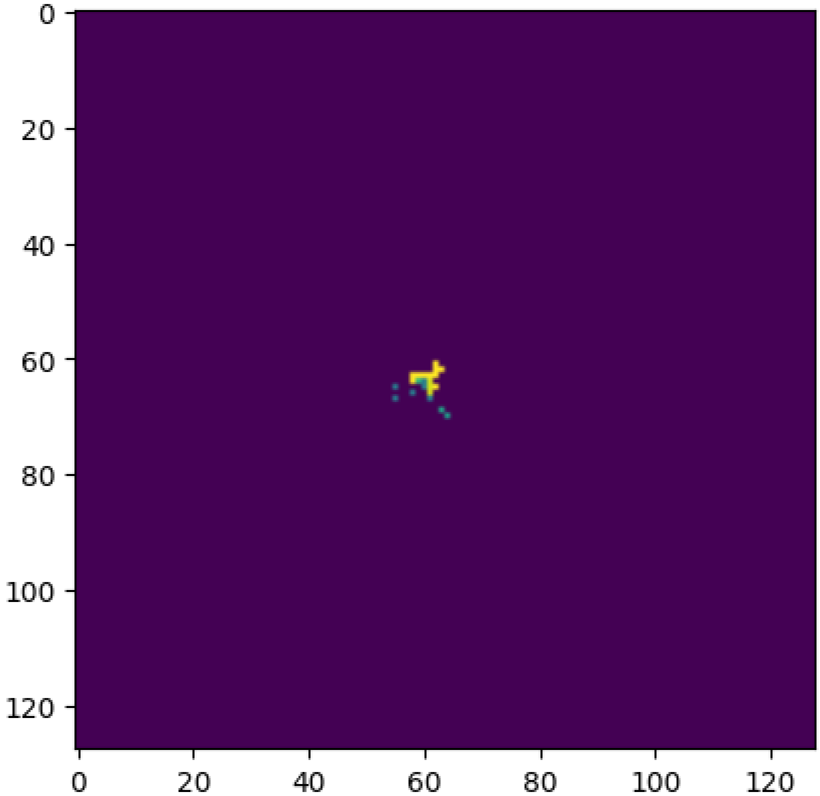
Out[18]:
```
(np.float32(0.57463396), np.float32(0.005351367))
```

In [19]:
```
splt_val = 0.15

for i in range(y_batch_np.shape[0]):
    plt.figure()
    plt.title(f"Actual {i + 1}")
    plt.imshow(y_batch_np[i, 0])

    plt.figure()
    plt.title(f"Predicted {i + 1}")
    val = y_pred_np[i, 0]
    plt.imshow(np.piecewise(val, [val < splt_val, val >= splt_val], [0, 1]))
    plt.imshow(val)
```

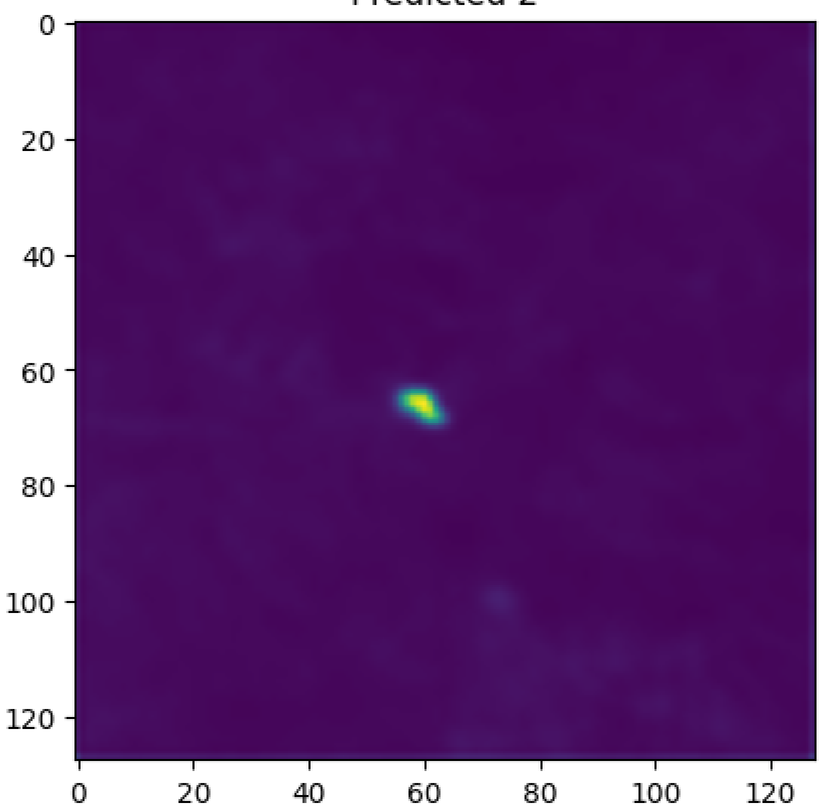

Actual 1
0
20
40
60
80
100
120
0 20 40 60 80 100 120

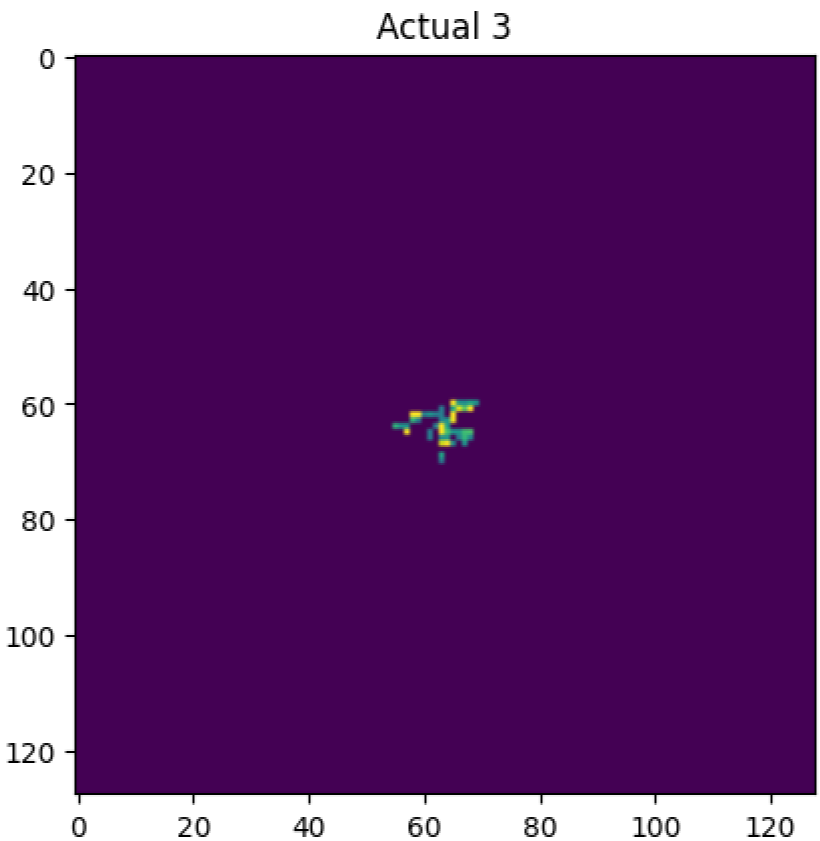

Predicted 1
0
20
40
60
80
100
120
0 20 40 60 80 100 120

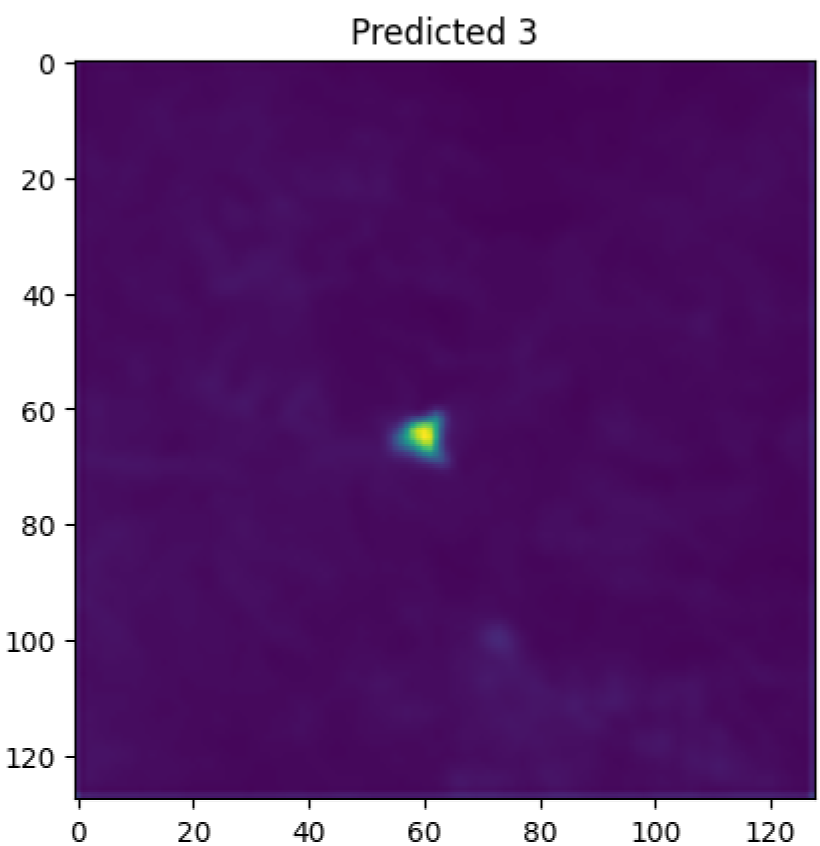
Actual 2

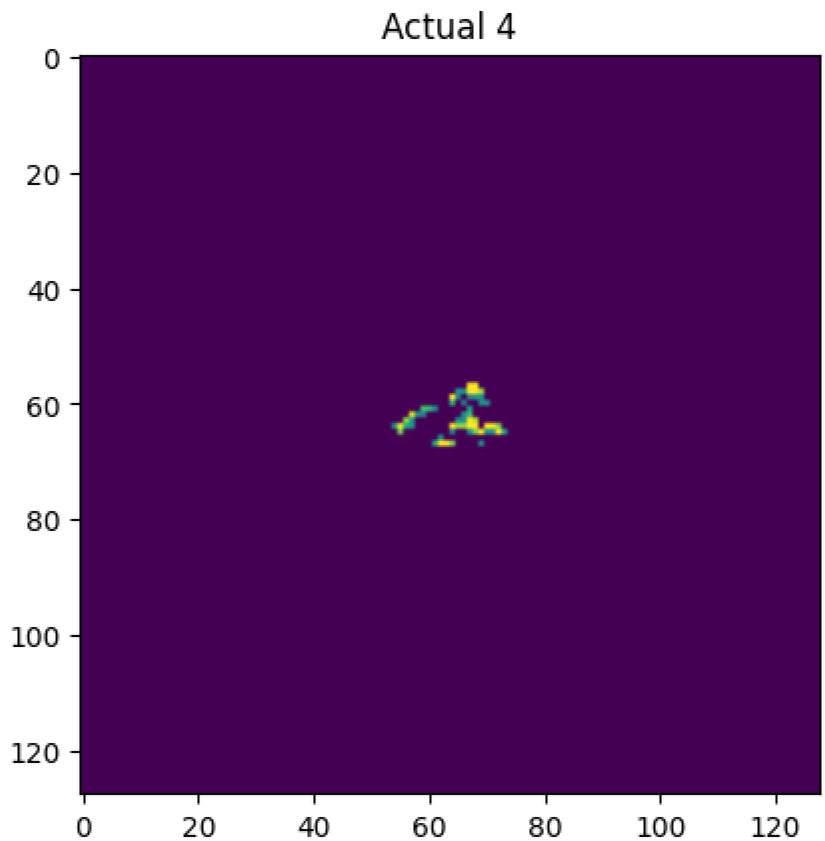


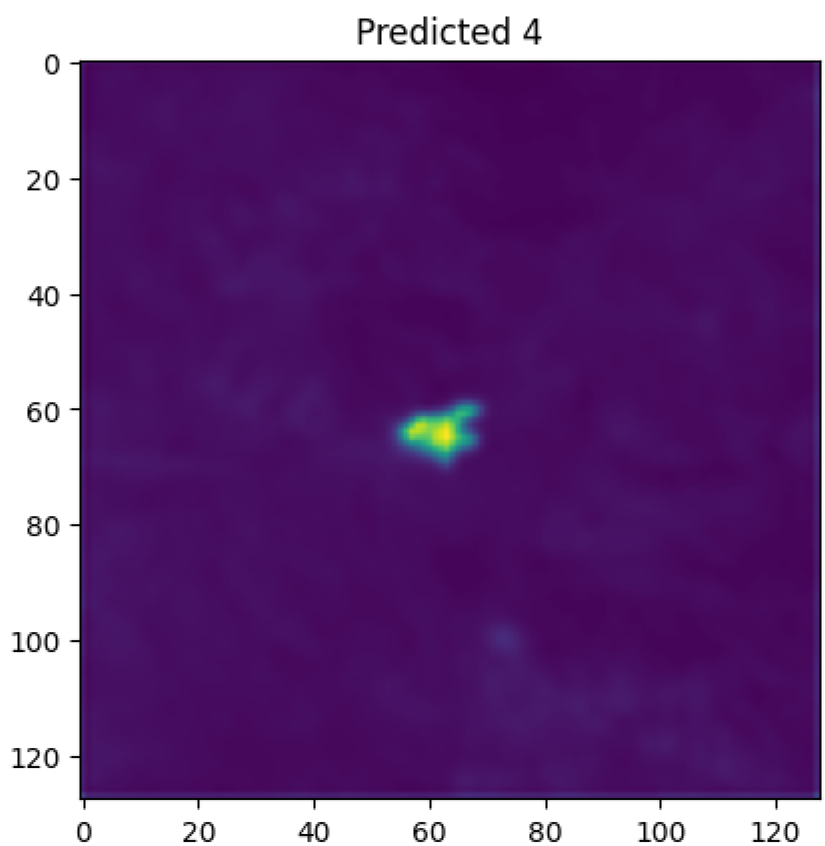
Predicted 2

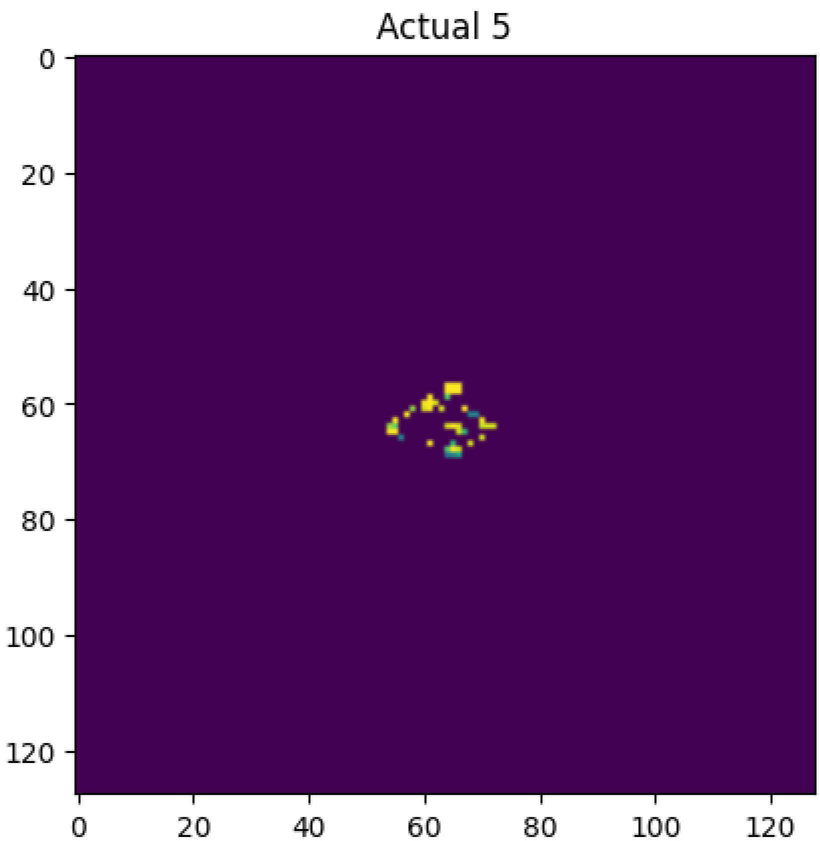


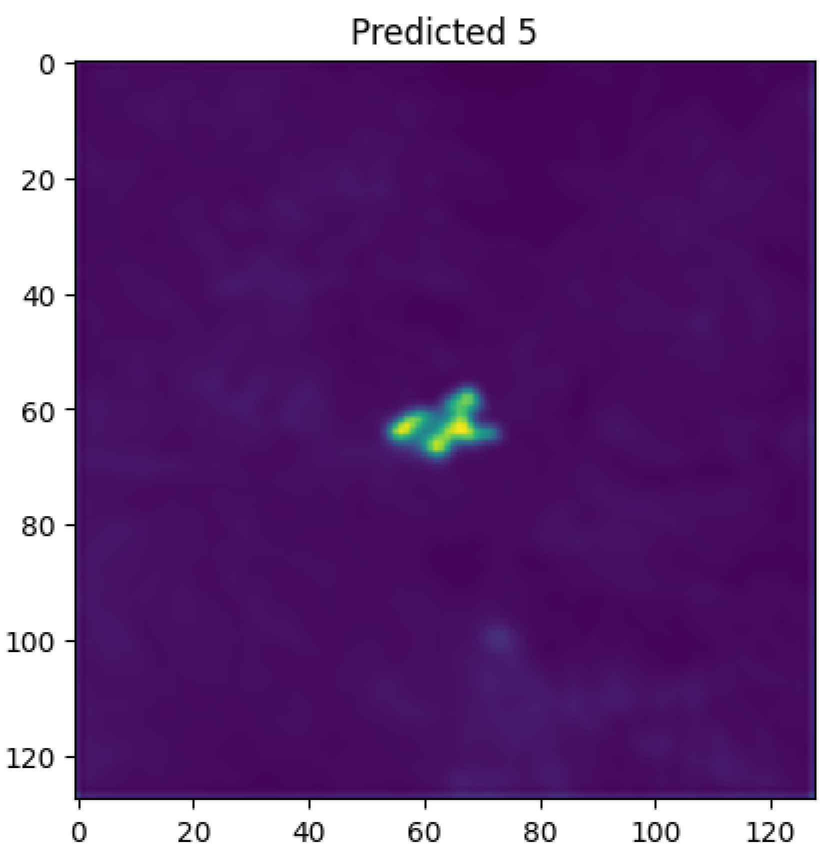

Actual 3
0
20
40
60
80
100
120
0 20 40 60 80 100 120


Predicted 3
0
20
40
60
80
100
120
0 20 40 60 80 100 120

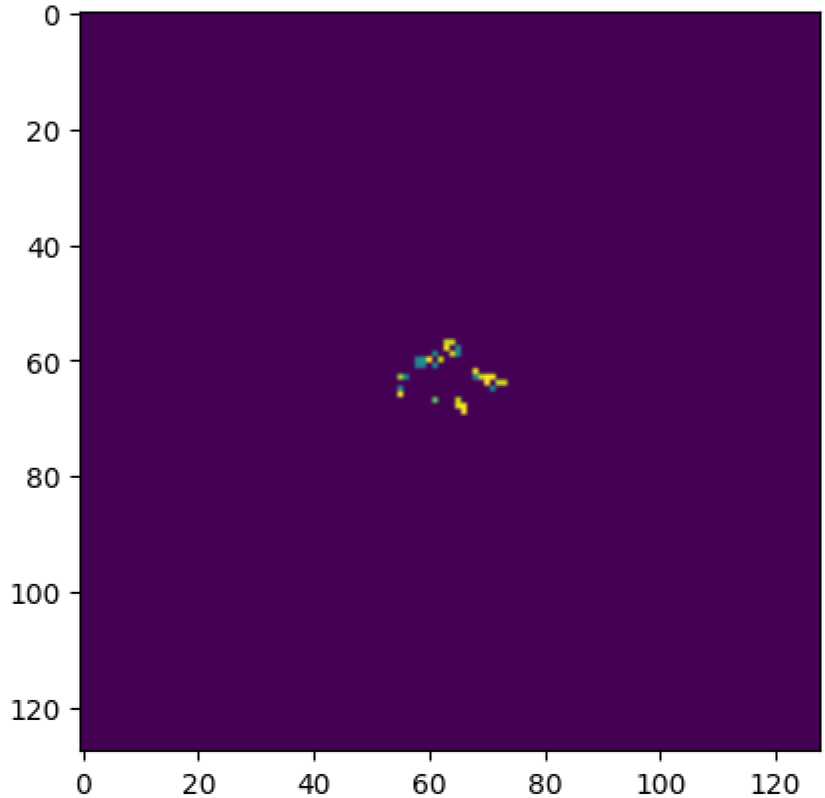

Actual 4
0
20
40
60
80
100
120
0
20
40
60
80
100
120


Predicted 4
0
20
40
60
80
100
120
0
20
40
60
80
100
120

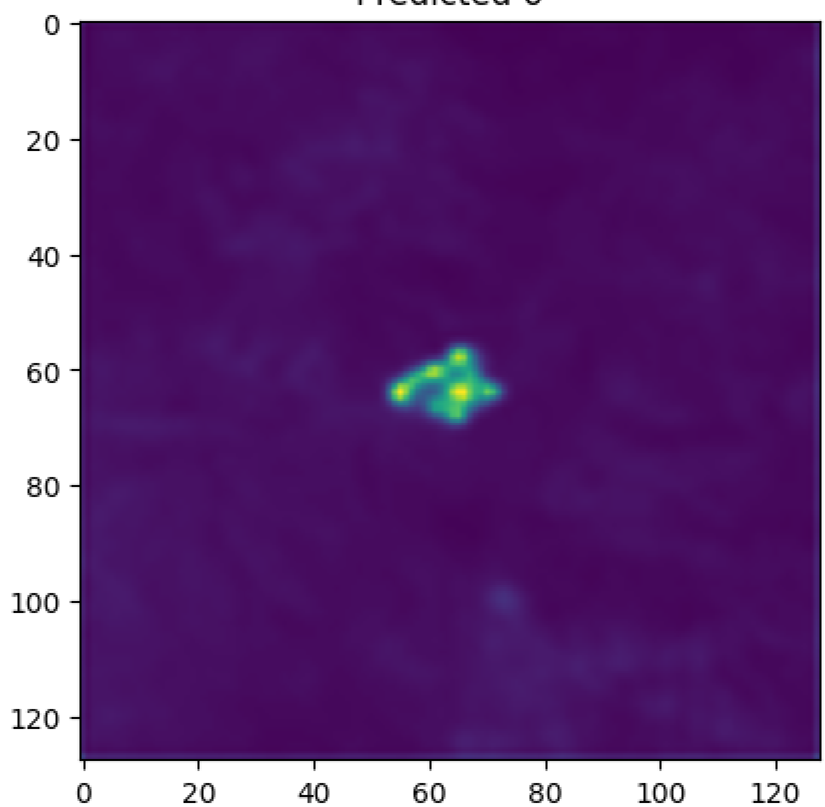

Actual 5
0
20
40
60
80
100
120
0 20 40 60 80 100 120


Predicted 5
0
20
40
60
80
100
120
0 20 40 60 80 100 120

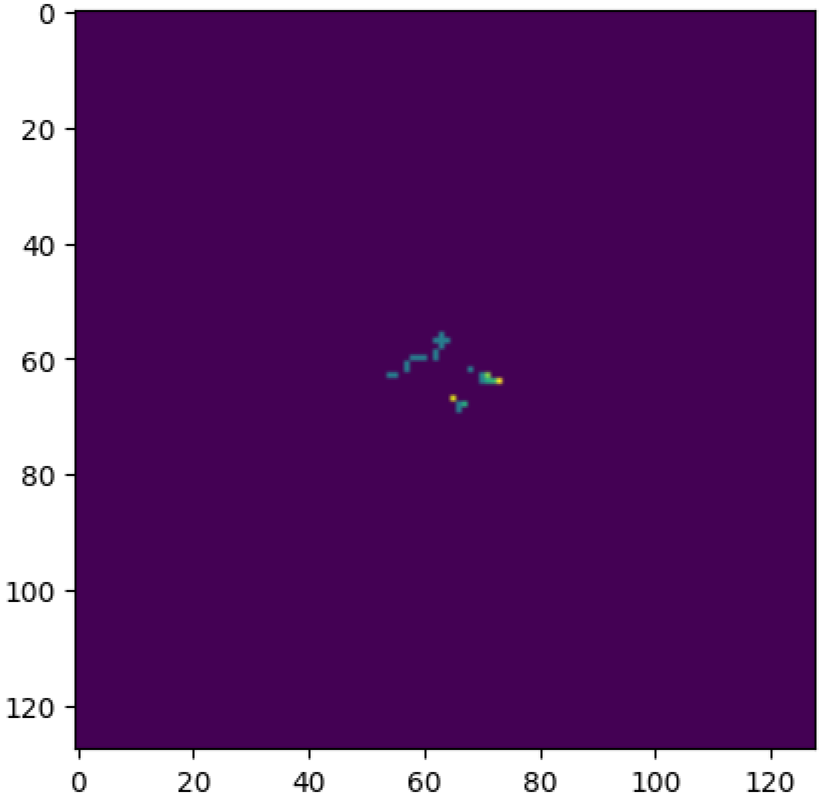
Actual 6



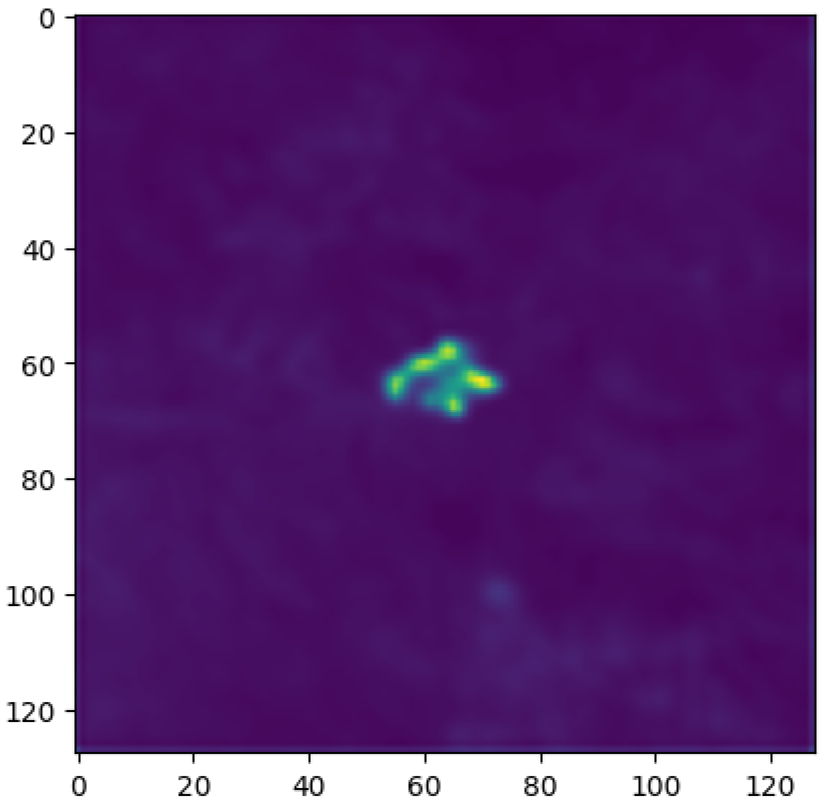
Predicted 6

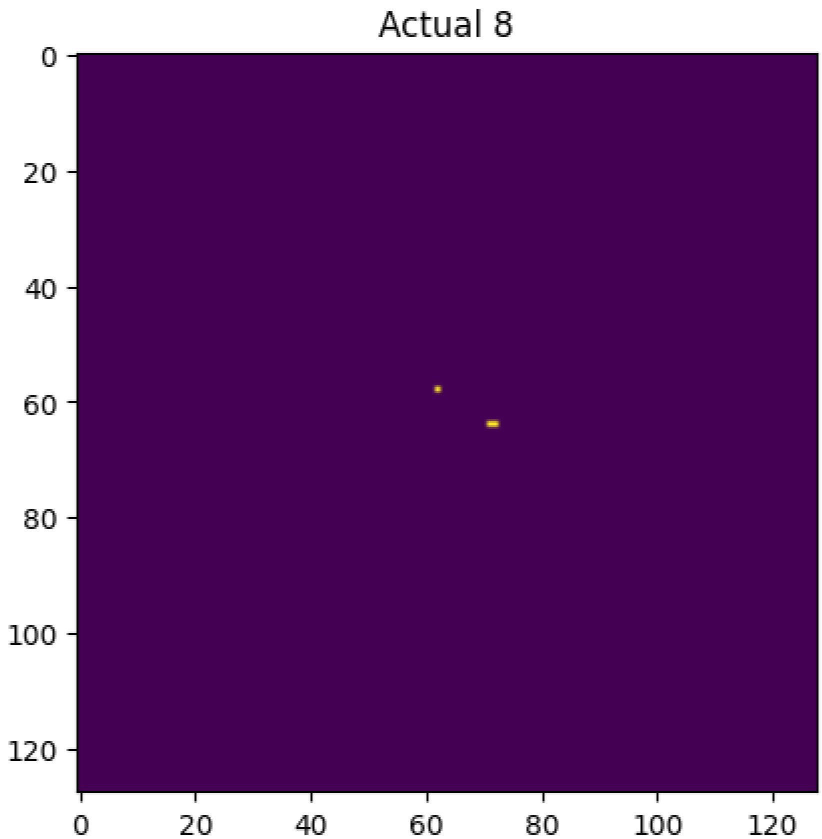


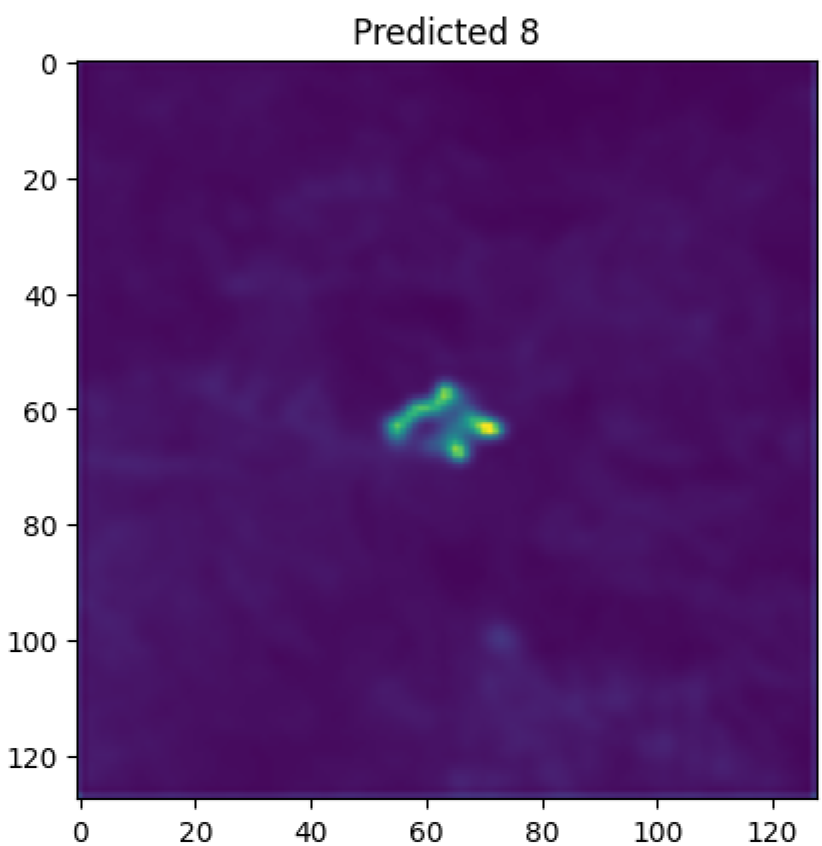
Actual 7



Predicted 7



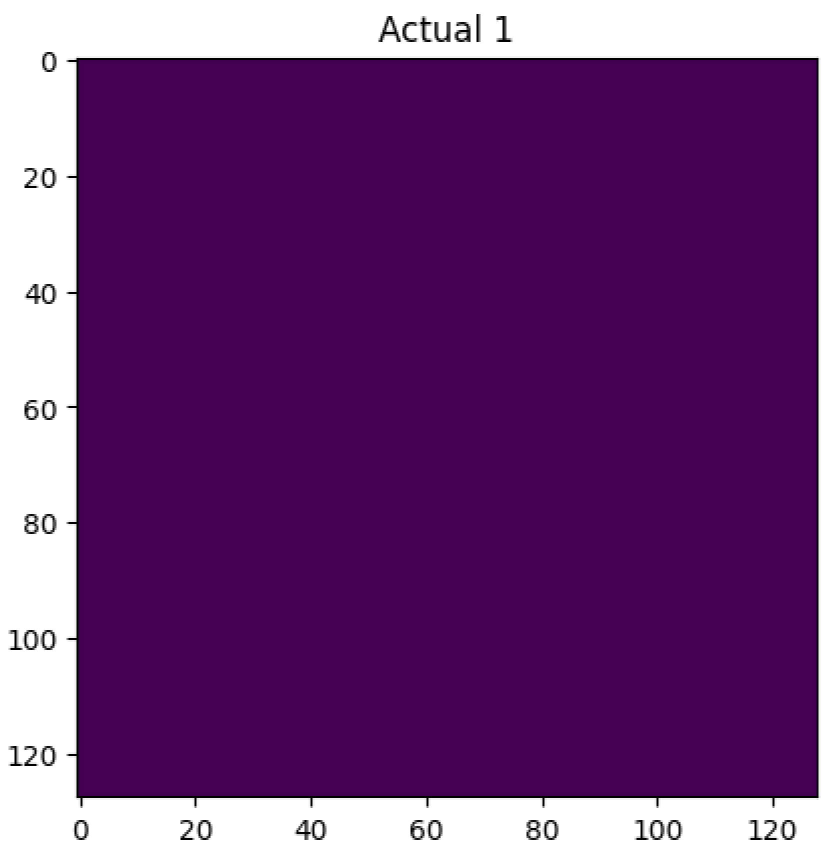


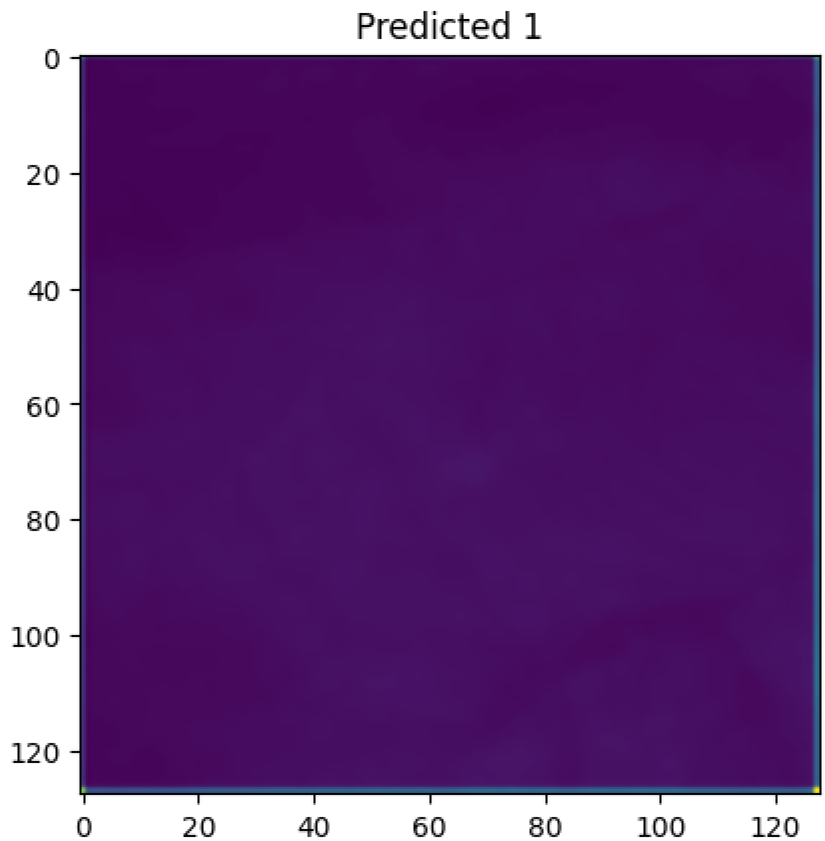


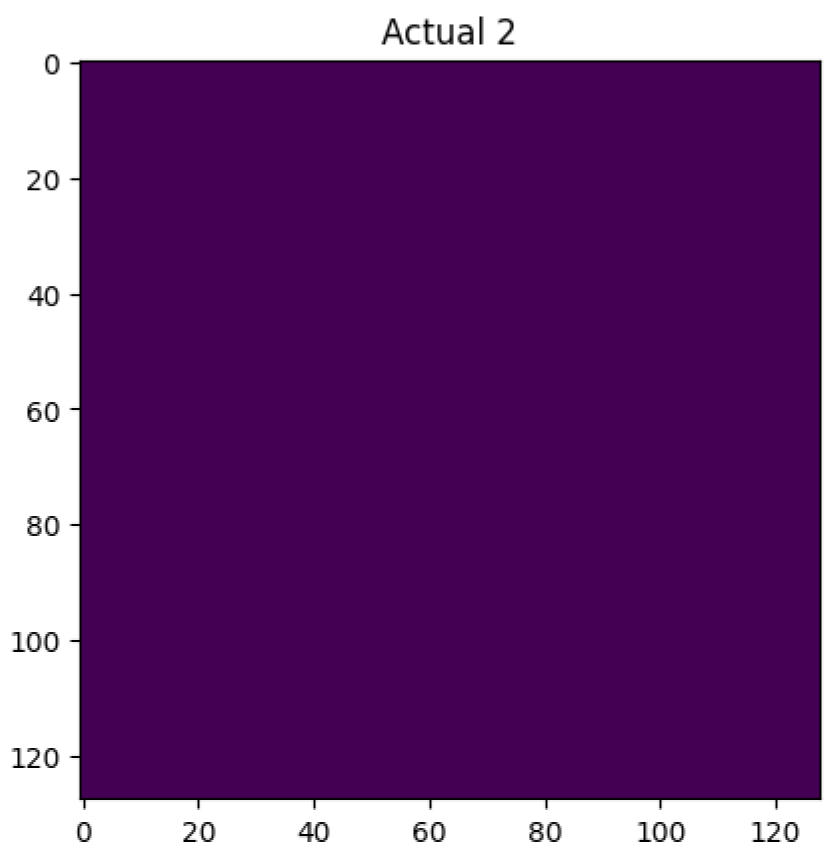
In [20]:
```
for batch_index, batch in enumerate(val_loader):
    if batch_index == 2:
        x_batch, y_batch = batch[0].to(device), batch[1].to(device)
```

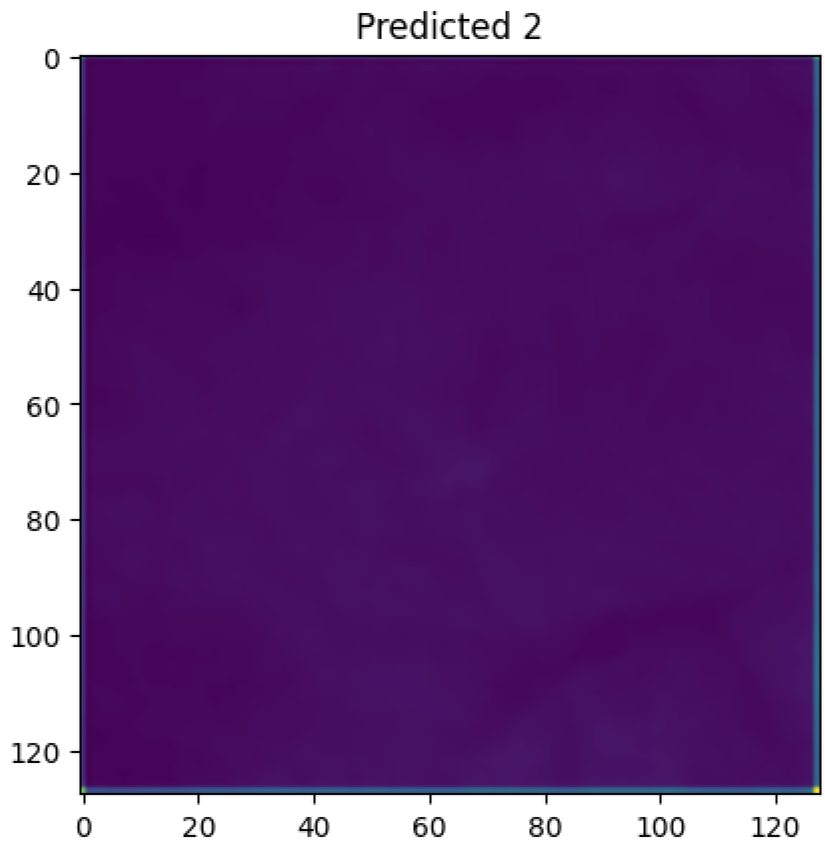
```
        y_pred = model(x_batch)
        print("found")
        break

print(torch.max(y_batch), torch.max(y_pred), torch.min(y_batch), torch.min(y_pred))

y_batch_np = y_batch.cpu().detach().numpy()
y_pred_np = y_pred.cpu().detach().numpy()

for i in range(y_batch_np.shape[0]):
    plt.figure()
    plt.title(f"Actual {i + 1}")
    plt.imshow(y_batch_np[i, 0])

    plt.figure()
    plt.title(f"Predicted {i + 1}")
    #plt.imshow(np.piecewise(y_pred_np[i, 0], [y_pred_np[i, 0] < 0.08, y_pred_np[i,
    plt.imshow(y_pred_np[i, 0])
```

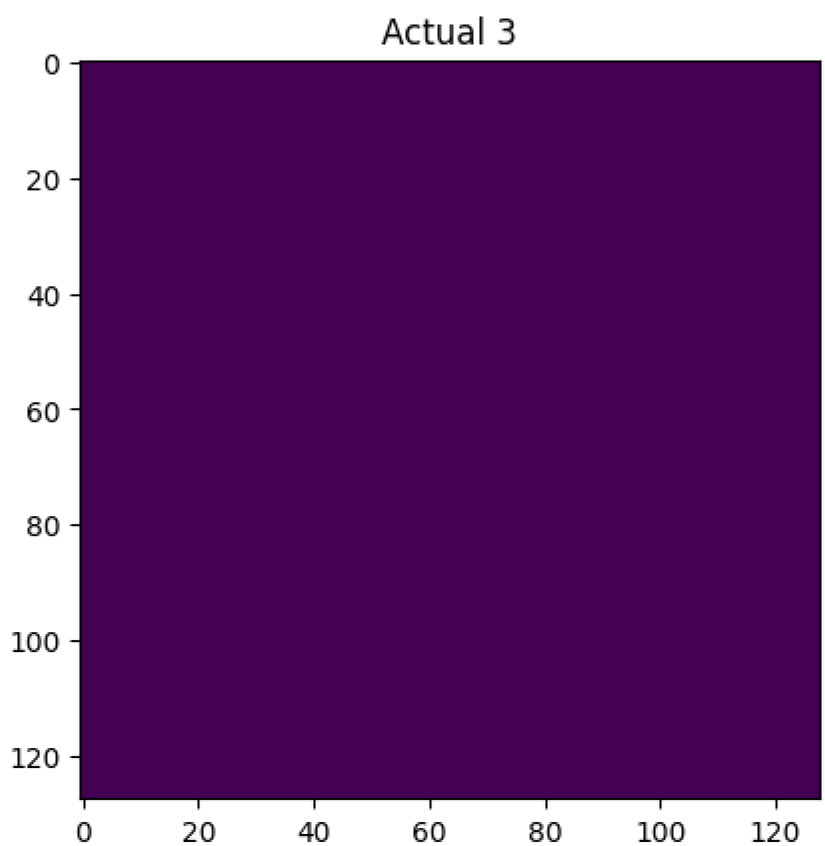
```
found
tensor(0.9098, device='cuda:0') tensor(0.2802, device='cuda:0', grad_fn=<MaxBackward
1>) tensor(0., device='cuda:0') tensor(0.0020, device='cuda:0', grad_fn=<MinBackward
1>)
```

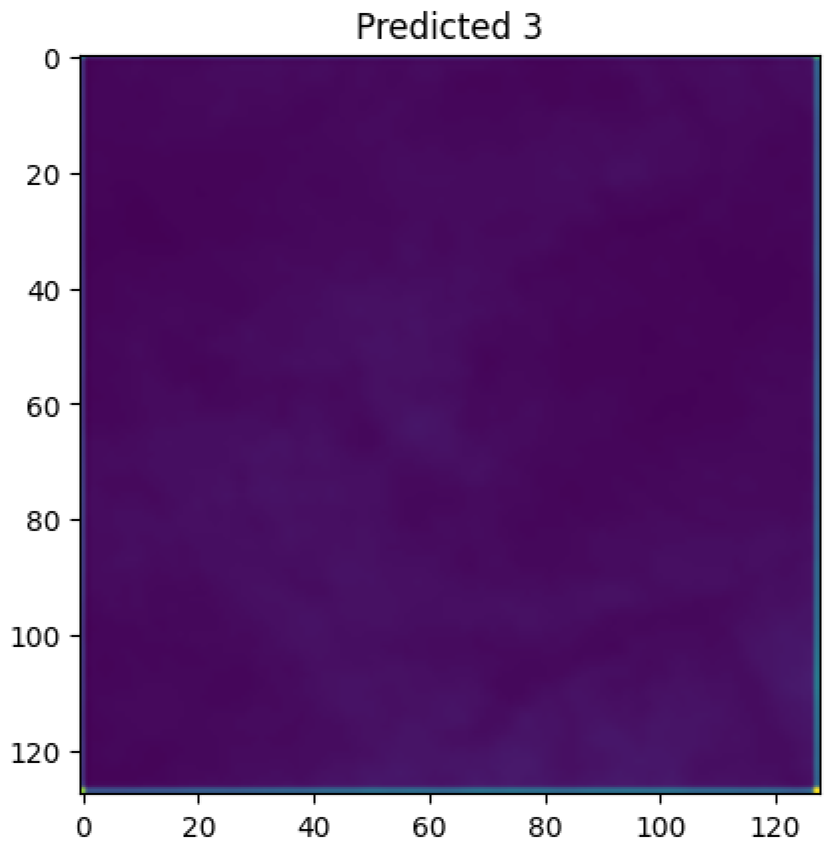


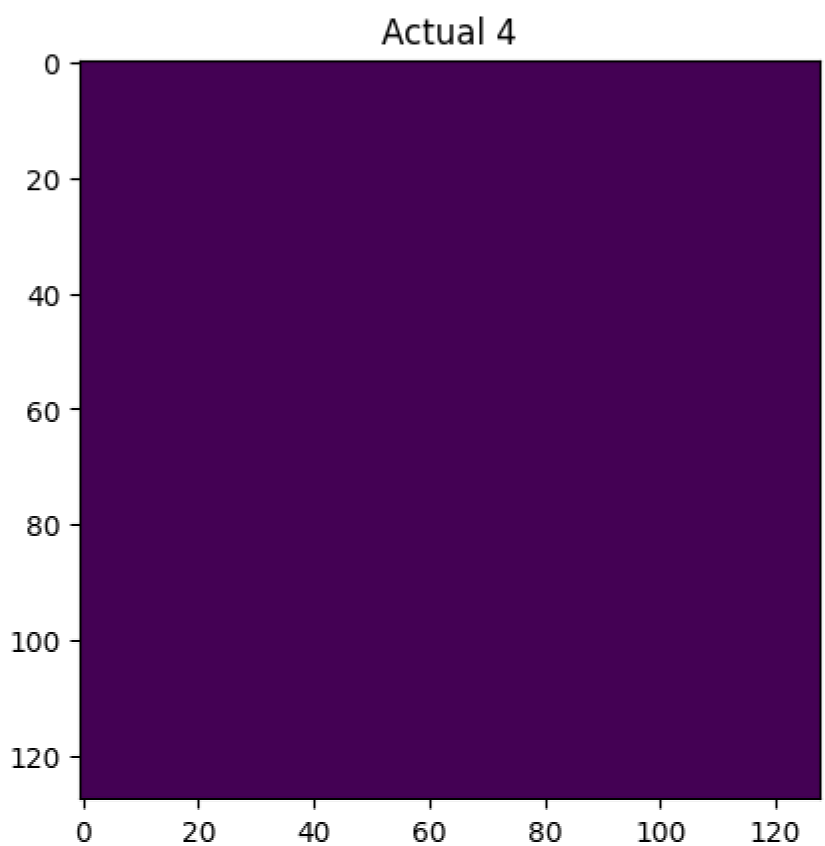

Predicted 1
0
20
40
60
80
100
120
0 20 40 60 80 100 120


Actual 2
0
20
40
60
80
100
120
0 20 40 60 80 100 120

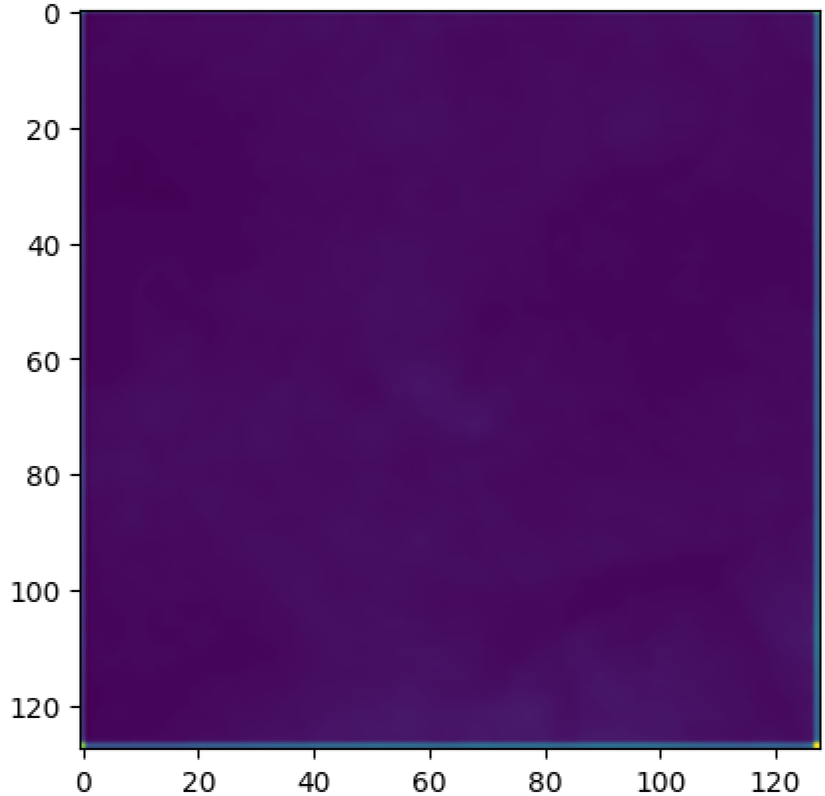

Predicted 2
0
20
40
60
80
100
120
0
20
40
60
80
100
120


Actual 3
0
20
40
60
80
100
120
0
20
40
60
80
100
120

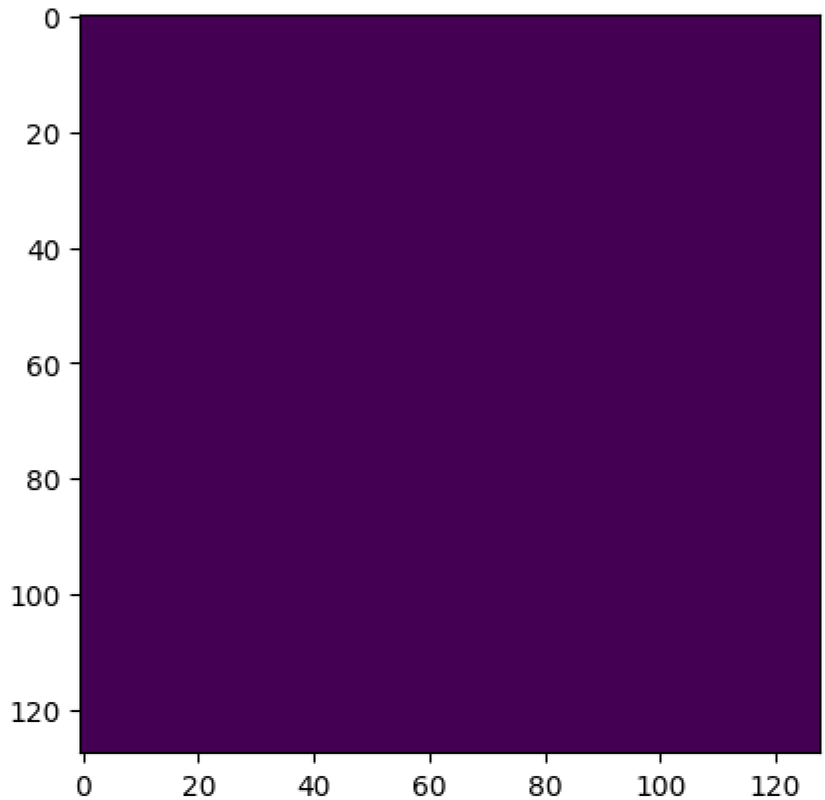

Predicted 3
0
20
40
60
80
100
120
0
20
40
60
80
100
120


Actual 4
0
20
40
60
80
100
120
0
20
40
60
80
100
120

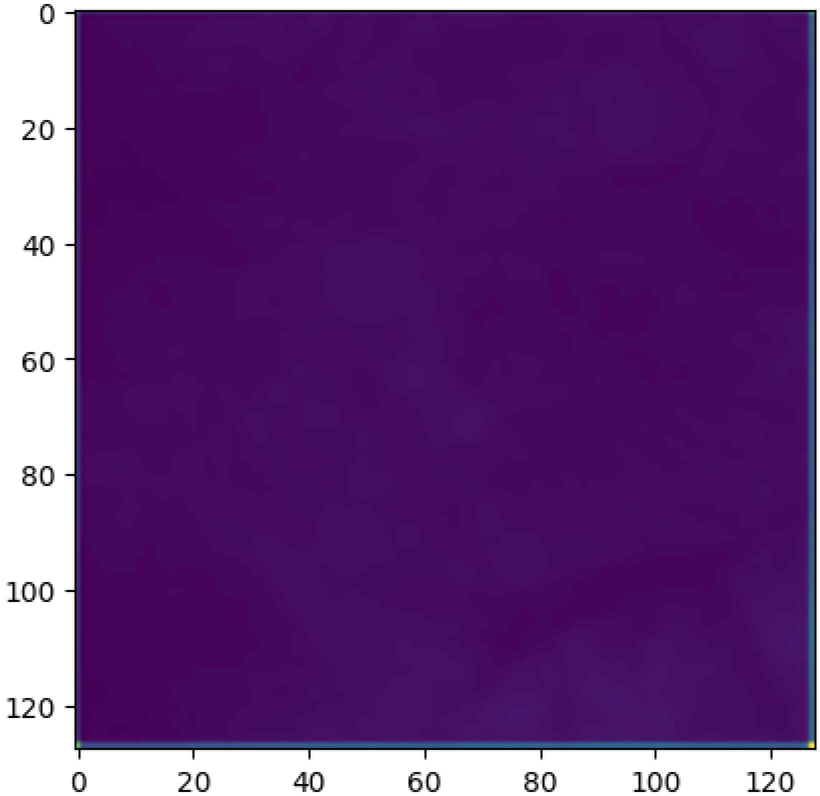
Predicted 4



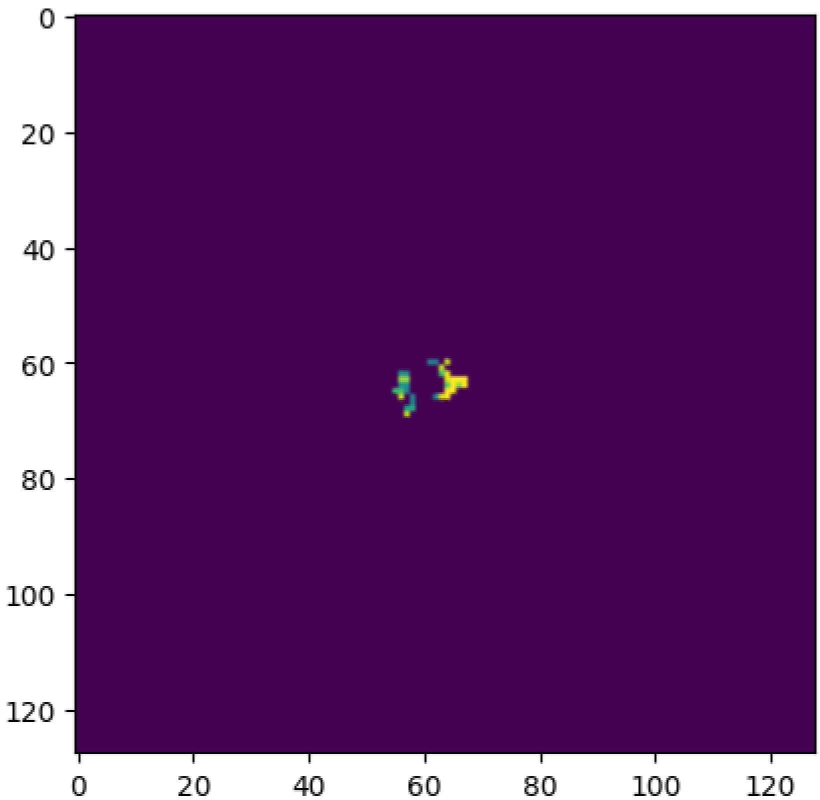
Actual 5

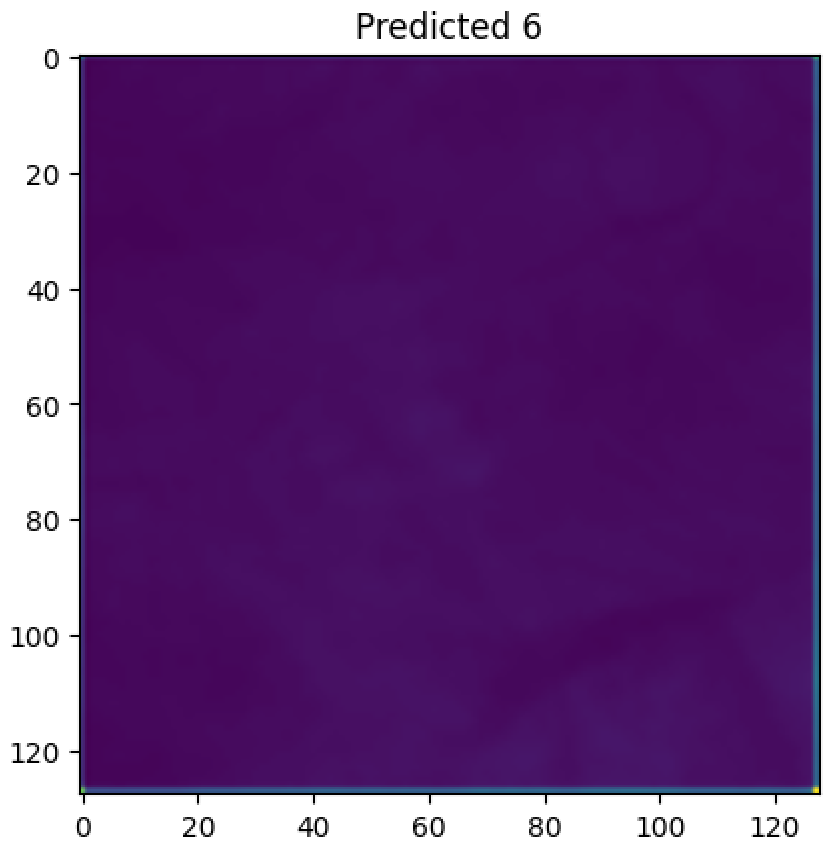


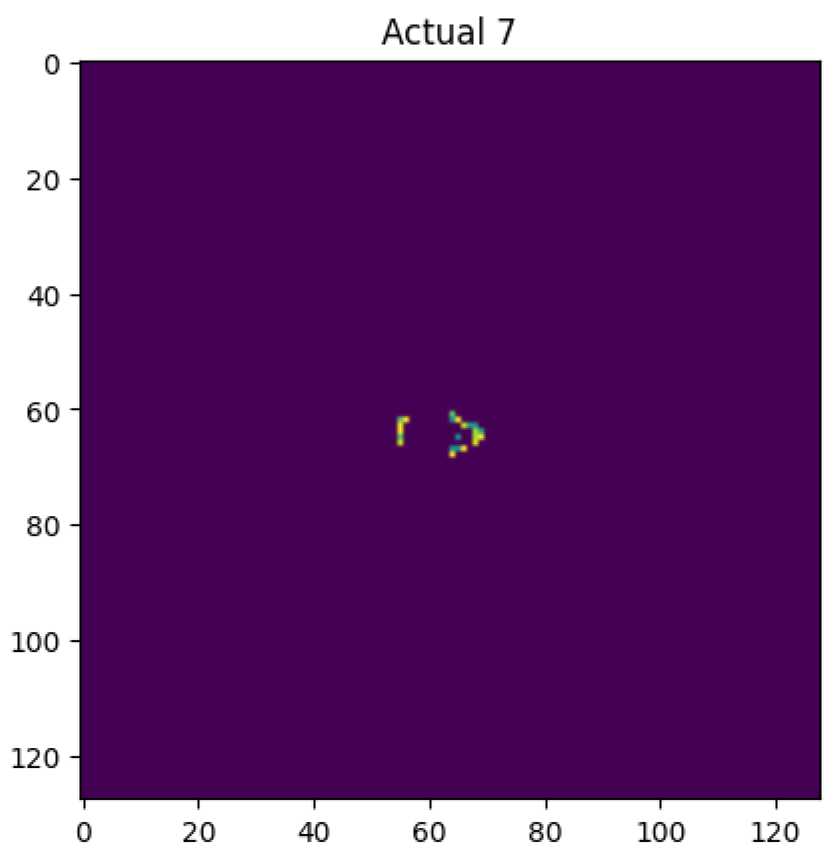
Predicted 5

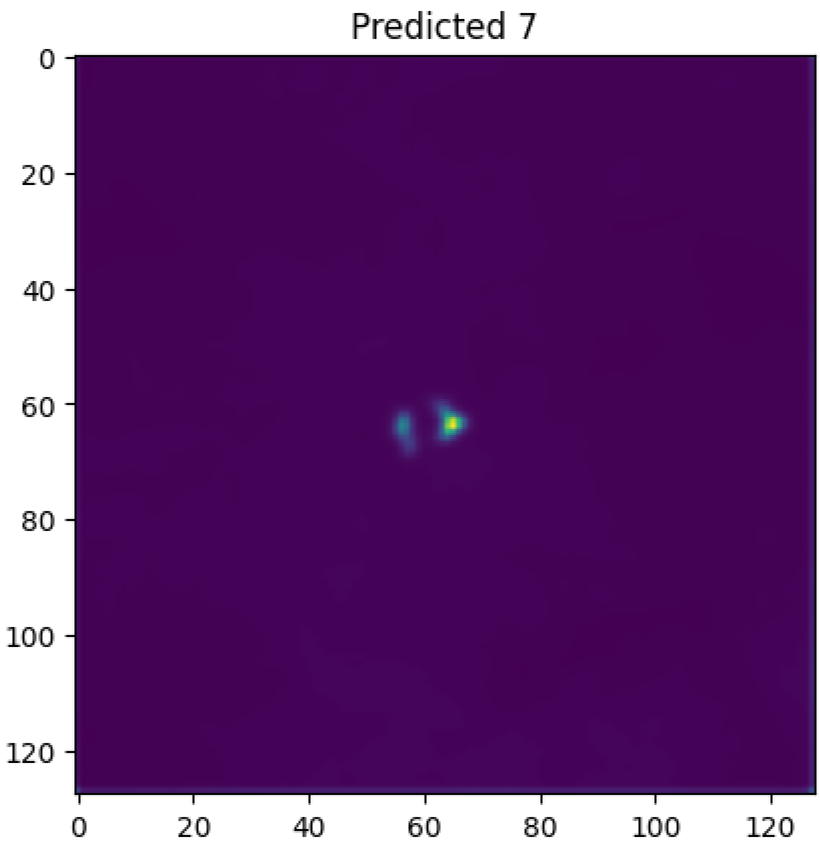


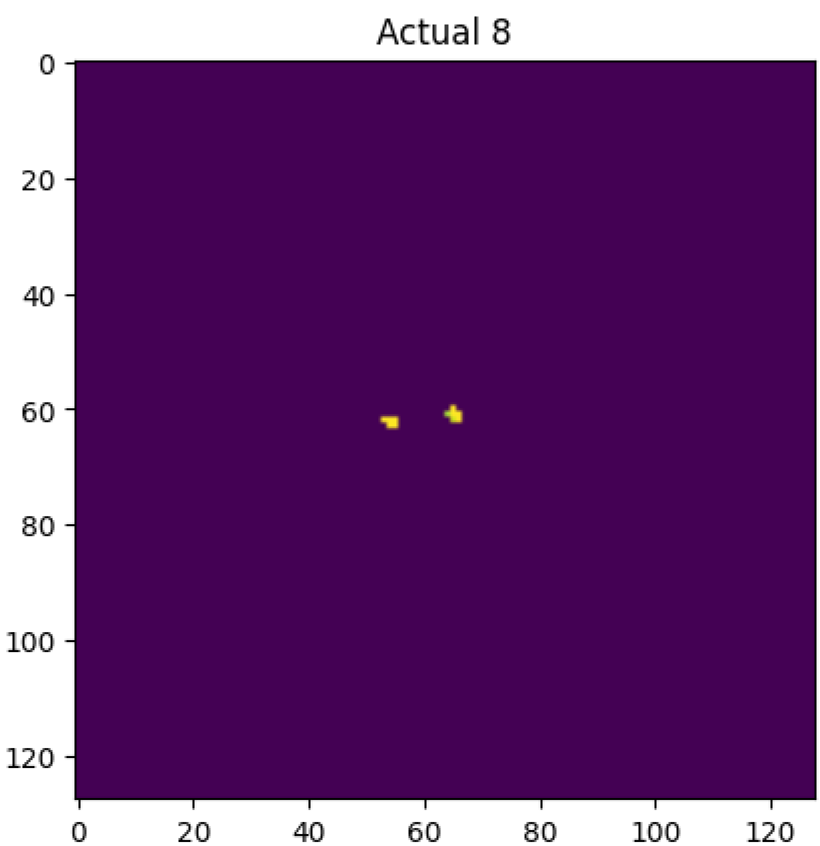
Actual 6

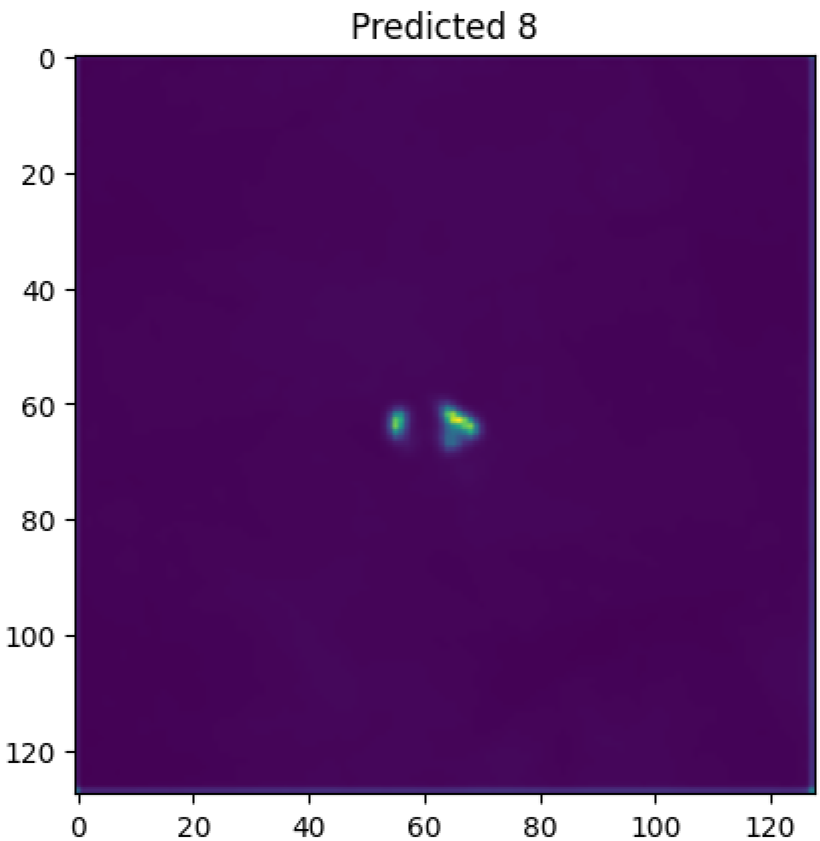



Predicted 6
0
20
40
60
80
100
120
0
20
40
60
80
100
120


Actual 7
0
20
40
60
80
100
120
0
20
40
60
80
100
120


Predicted 7
0
20
40
60
80
100
120
0 20 40 60 80 100 120


Actual 8
0
20
40
60
80
100
120
0 20 40 60 80 100 120




```
In [ ]:
```