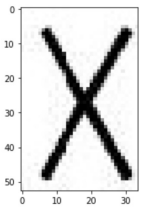
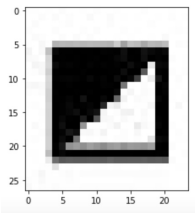
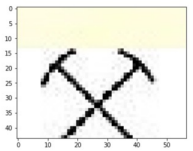
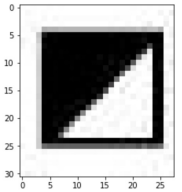
Outline

- Validation result feedbacks
- Current method

Validation result feedbacks

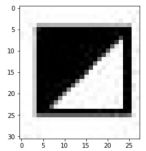
- Leaderboard score: 0.02043687 (11/02)
  - # total predictions: 212
  - # submitted: 140
- Feedback for each map:

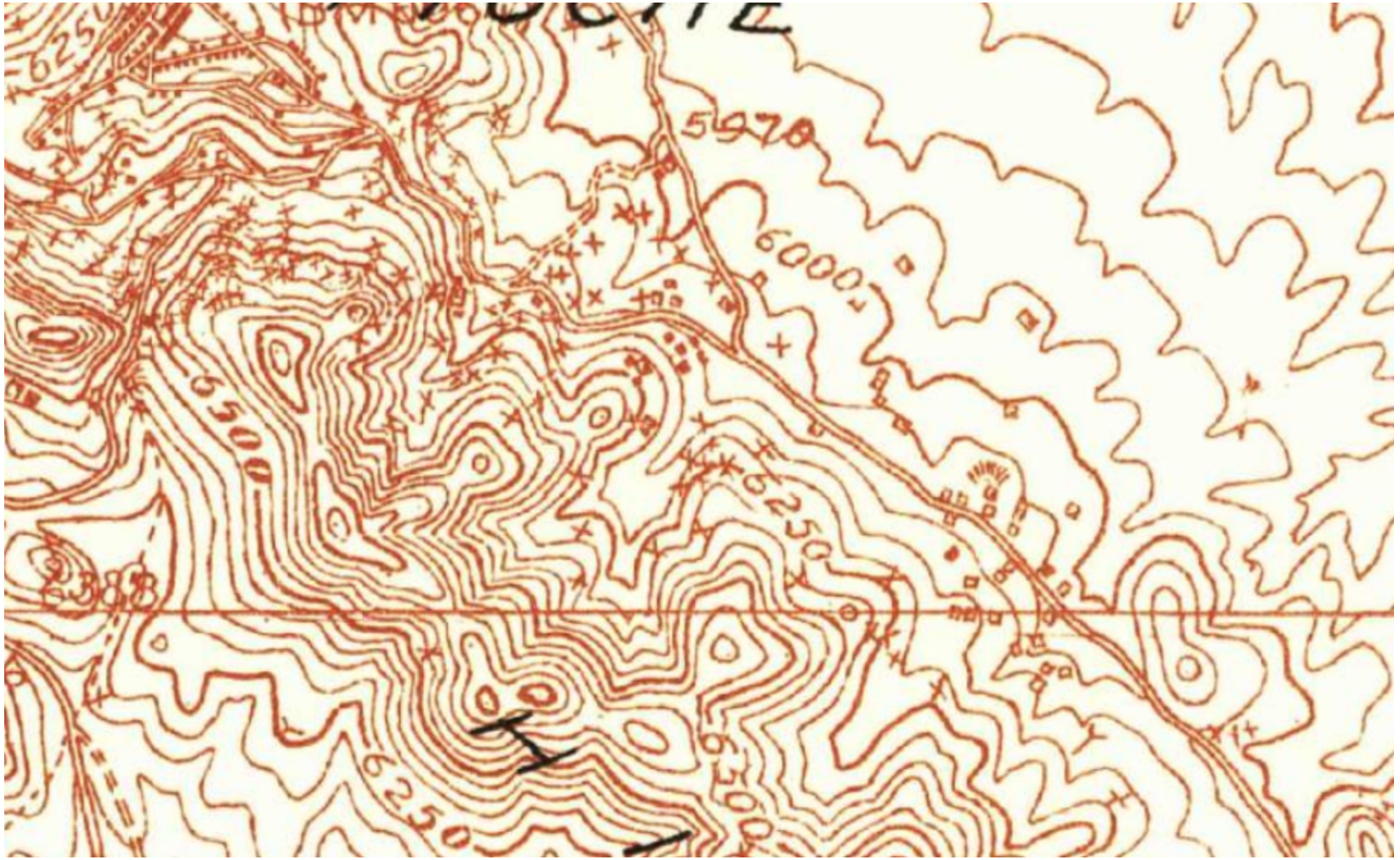
A	B
raster_file	f_score
NM_SantaRita_192296_1951_24000_geo_mosaic_3_pt.tif	0.5267052249
CO_Silverton_402615_1955_62500_geo_mosaic_2_pt.tif	0.5700494412
CA_Coulterville_297201_1947_62500_geo_mosaic_5_pt.tif	0.04185202542
NV_ElyRange_320661_1921_48000_geo_mosaic_2_pt.tif	0.03978368386

Validation result feedbacks

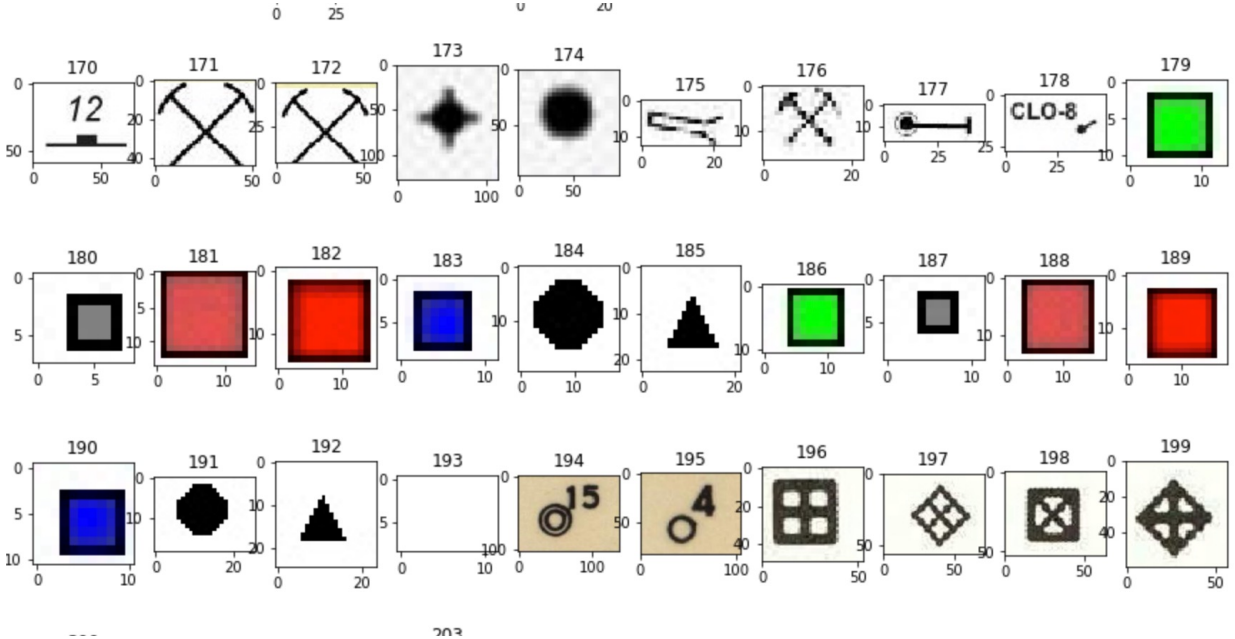
- NV_ElyRange



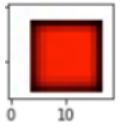


Method: Assembly model

- Motivation:
  - Some symbols can be distinguished by colors while others by shapes
- Idea: Assemble three models, first one is color based (CV model), second one is shape based (DNN model) and third one is shape based template matching

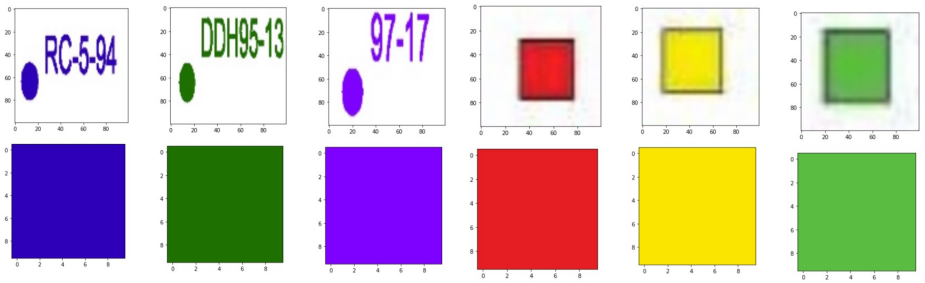


Method Color-based

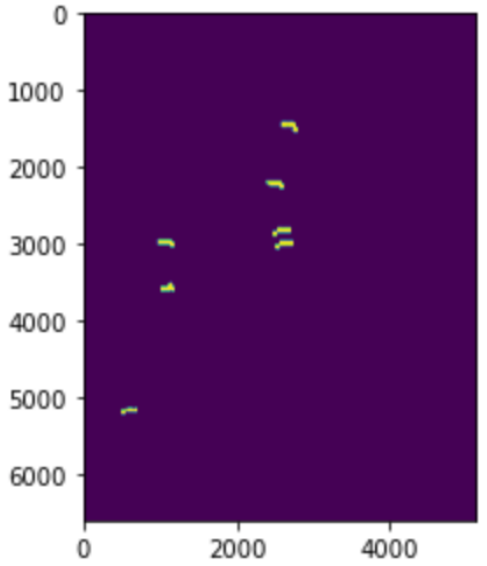
- Idea:
  - Find the symbol locations by searching for similar RGB values as template
- Challenges:
  - Decide the color of the template 
    - (remove black border, deal with uneven color)
  - Output needs to be a point location instead of segmentation mask

Method Color-based

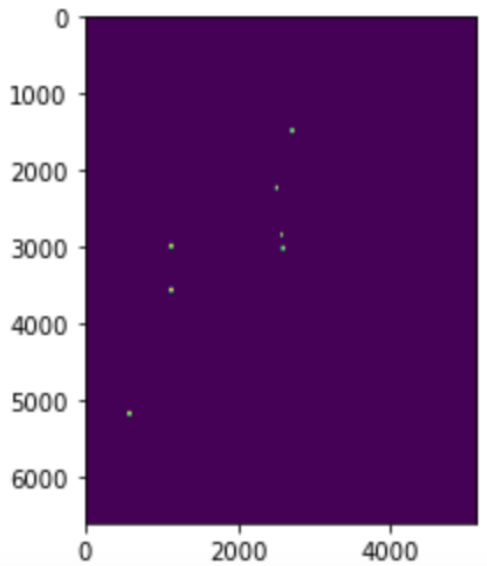
- Method:
  - On template image, use adaptive binarization to select the colored region
  - Calculate the mode color for colored region
  - With RGB color range 20, search for pixels with similar color on the whole map
  - On the segmentation mask, apply morphological operation to remove noise
  - Find the contours on seg. mask
  - Obtain mass center of contours



mode color samples



segmentation mask with color searching



mass center of contour

Method DNN - Training

- Idea: Train a NN classifier for **each** point symbol
- Training Data Preparation:
  - All samples are cropped from the foreground region
    - background: pure white color
  - **Positive** samples:
    - Crop around the ground-truth location. Crop size is the same as the template symbol size
  - **Negative** samples:
    - **Hard negative: (p=0.25)** other symbols in the same map
    - **Random negative: (p=0.75)** randomly crop from foreground region which does not overlap with positive samples

Method DNN - Training

- Augmentation:
  - Translation and scaling
  - Resize to 64x64
  - Normalization with mean=0.5, std=0.5

  - [Update since last submission]: Rotation augmentation for some symbols

```
# 'fix':['x','triangular_matrix','quarry_open_pit','crossed_downward_arrows','button','triangle','dot','c_dot',
'solid_colored_circle','asterix','purple_arrow_kite','diamond_words']

# 'rotate':['sleeping_y','small_inclined_fault_num','plus','reverse_p_num','barbeque_tofu','fault_line_triangle_num',
'line_diamond_center_solid','small_inclined_fault','inclined_fault',
# 'small_inclined_triangle_fault_num','christmas_tree','fault_line_triangle_hollow_num','line_diamond_center',
'arrow_circle','arrow_num',]
```

Method DNN - Training

- Model for last submission: 8 models

```
index_list_dict = {
    'x' : [22,37,46,47,64,58,62,66,69,109,114,121,125,129,132,137,141,145,149,153,158,
        162,164,],
    'triangular_matrix': [21,36,44,57,61,65,68,75,108,113,124,128,131,136,140,
                          144,148,152,157,161,163,],

    'quarry_open_pit' : [24,39,60,64,107,111,116,122,127,134,139,143,155,160,166,171,172,
                         176,206,208,210,212],
    'crossed_downward_arrows' : [20,56,123,135,147,151,156,204,207,209,211,],
    'button': [26,35, 48,119,168 ], # 35?
    'triangle' : [28, 185,192],
    'dot':[9,31,174,191,],
    'sleeping_y':[23,38,46,55,59,63,67,110,115,126,130,133,138,142,154,159,
                  165,205,]
}
```

Method DNN- Training

- Network 1:
  - Input:
    - **64x64x3** image patch
  - Output:
    - Prob of being target symbol or not

```
class Net64(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(3, 6, 5)
        self.pool = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(6, 16, 5)
        self.conv3 = nn.Conv2d(16, 32, 5)
        self.fc1 = nn.Linear(32 * 4 * 4, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 2)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = self.pool(F.relu(self.conv3(x)))

        x = torch.flatten(x, 1) # flatten all dimensions except batch
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        # print(x.shape)
        return x
```

Method DNN- Training

- Network 2:
  - Input:
    - **128x128x3** image patch
  - Output:
    - Prob of being target symbol or not

```
class Net128(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(3, 32, 5)
        self.pool = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(32, 128, 5)
        self.conv3 = nn.Conv2d(128, 64, 5)
        self.conv4 = nn.Conv2d(64, 16, 5)
        self.fc1 = nn.Linear(16 * 4 * 4, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 2)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = self.pool(F.relu(self.conv3(x)))
        x = self.pool(F.relu(self.conv4(x)))

        x = torch.flatten(x, 1) # flatten all dimensions except batch
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        # print(x.shape)
        return x
```

Method DNN - Inference

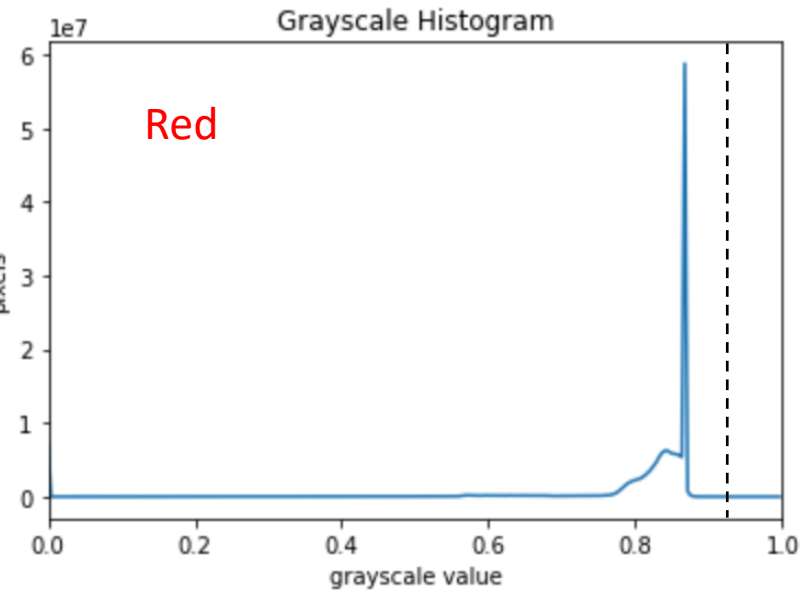
- Obtain template image and get the template size
- Crop the map image with overlapping ratio 0.5 (half of template size)
- Apply trained model on cropped patches and make prediction

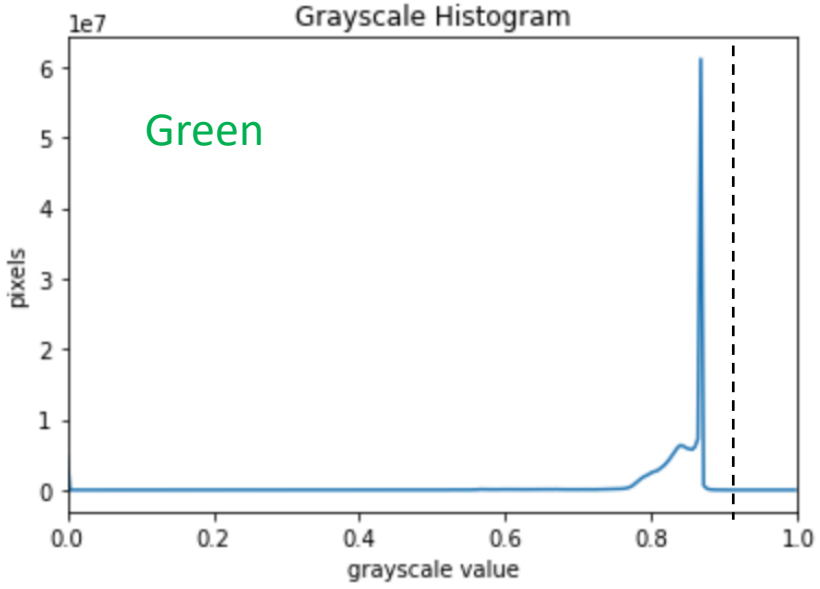
Method Template Matching

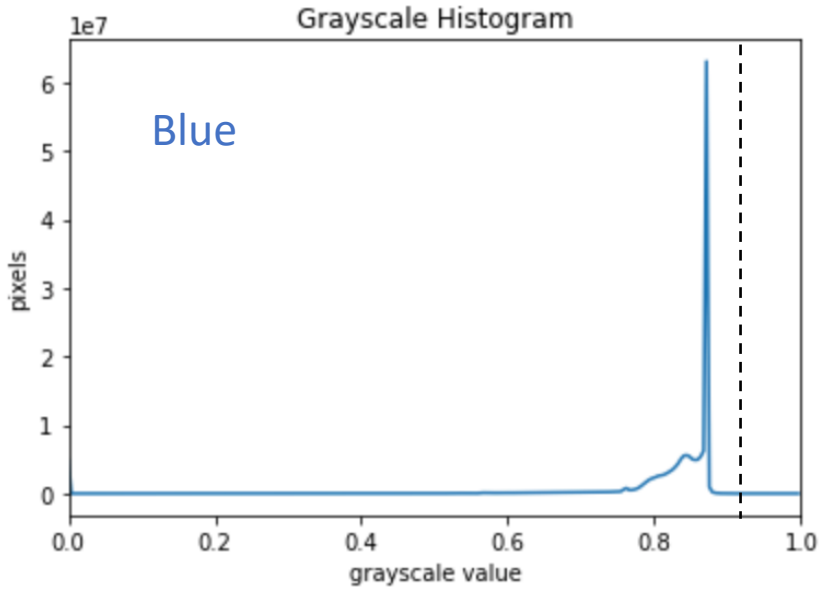
- Steps:
  - Split the image into R,G,B channels
  - Perform OpenCV template matching with TM_CCORR_NORMED method for each channel
    - Automatically determine the similarity threshold by choosing the elbow point
  - Find the intersection of three channels that have high similarity to template image

Method Template Matching

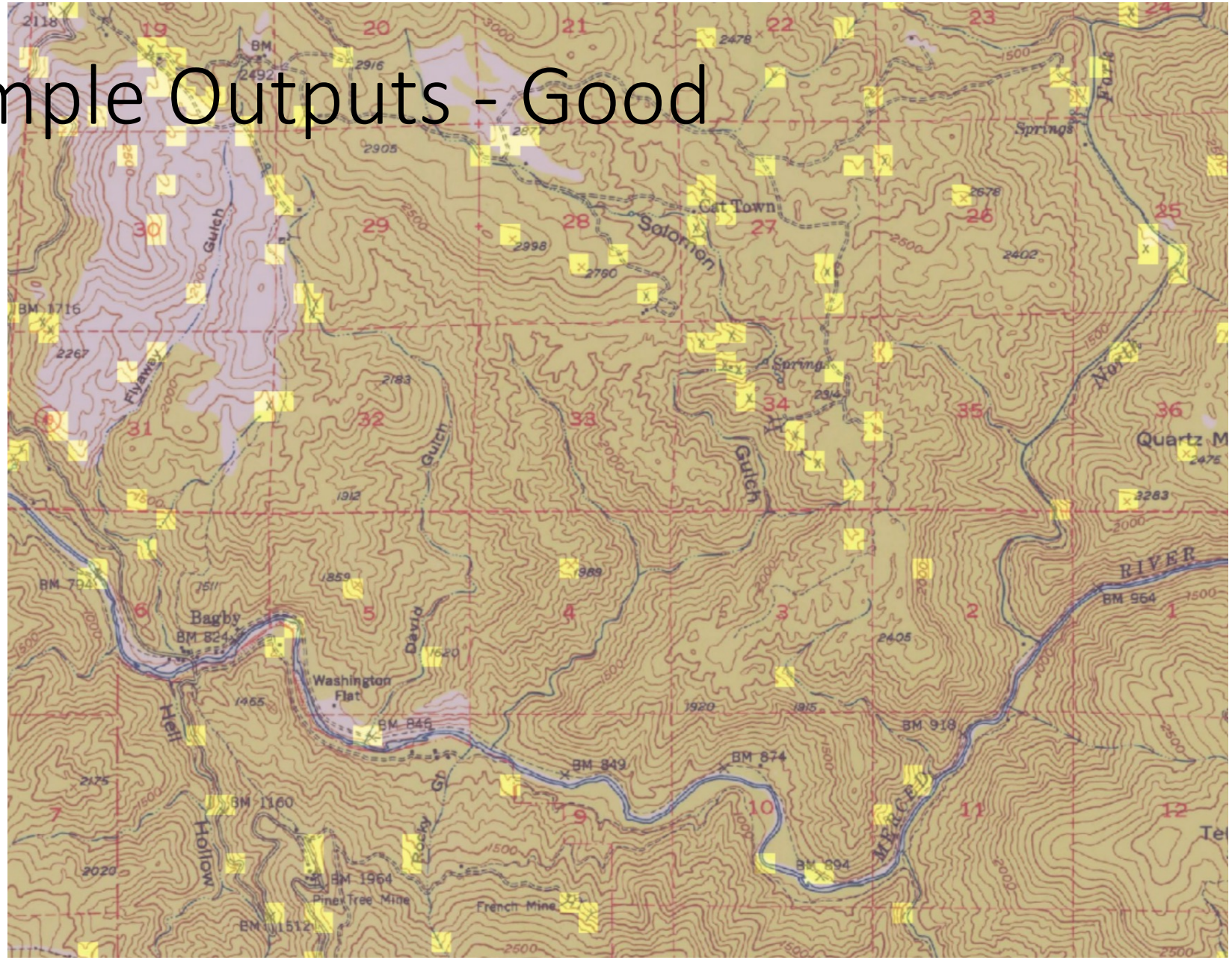
- Determine similarity threshold:
  - Rescale grayscale from [0,255] to [0,1]
  - Construct the grayscale histogram in R, G, B channel respectively
  - Select the threshold value to be slightly larger than the elbow gray scale value



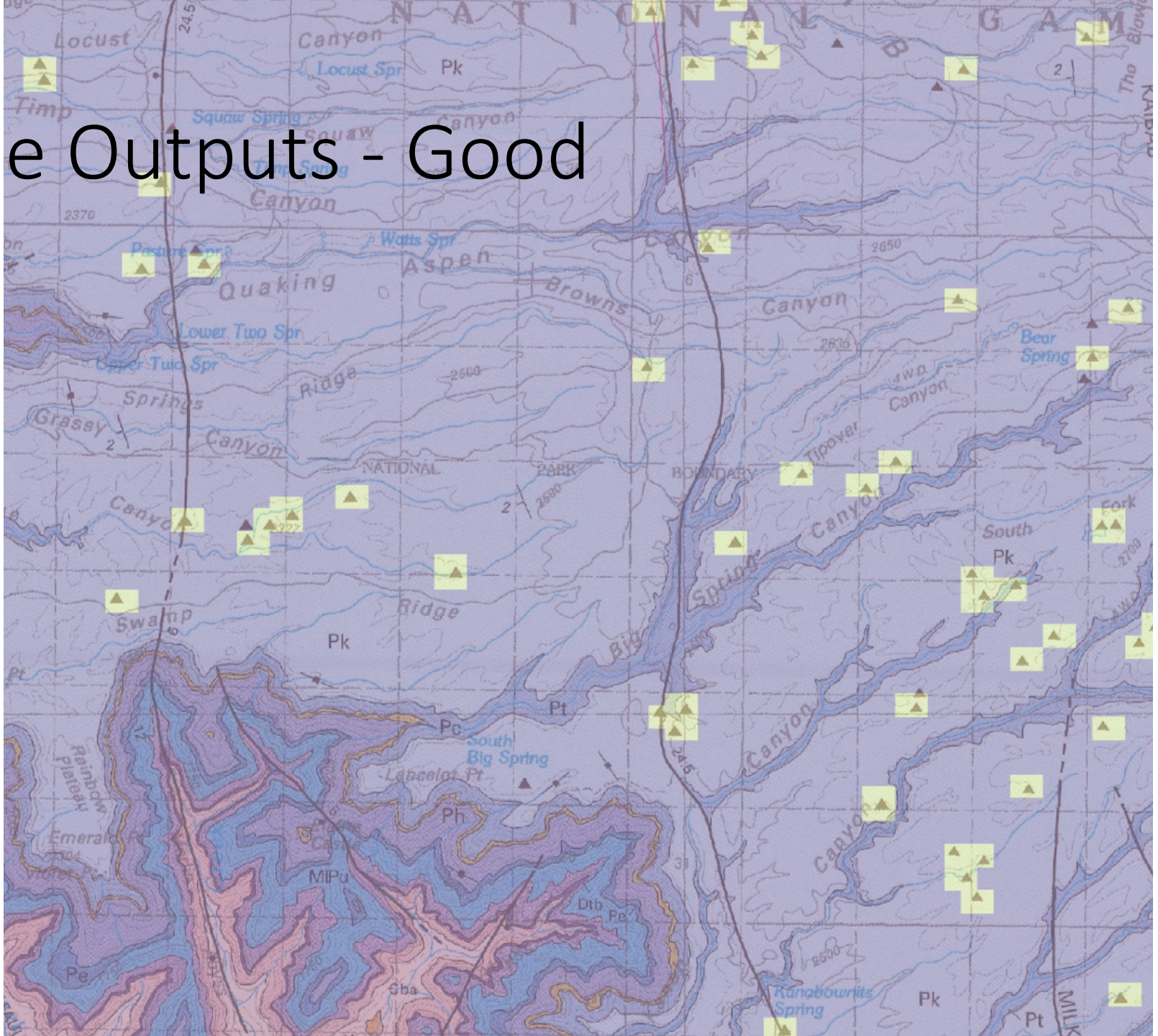




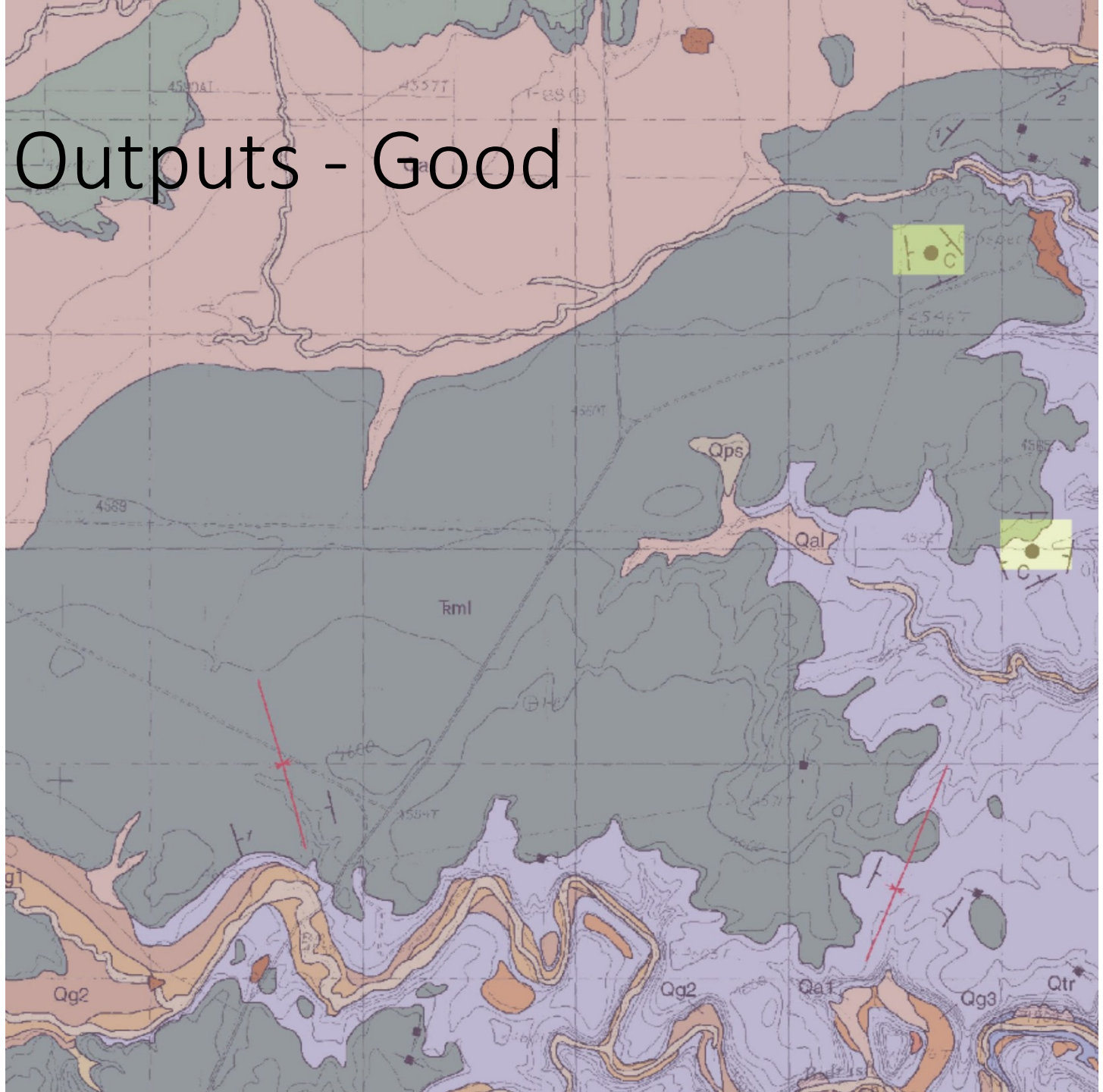
Sample Outputs - Good



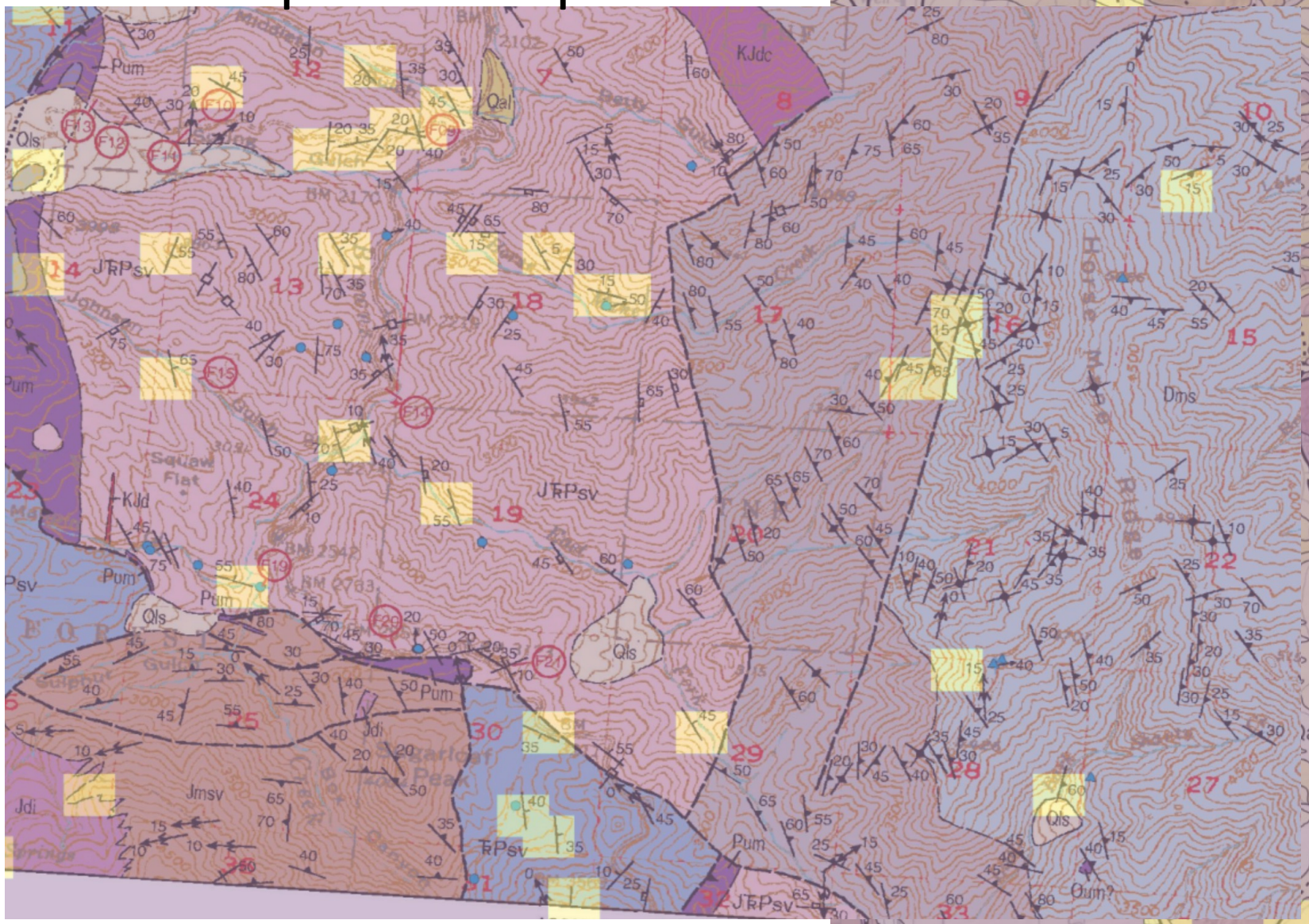
Sample Outputs - Good

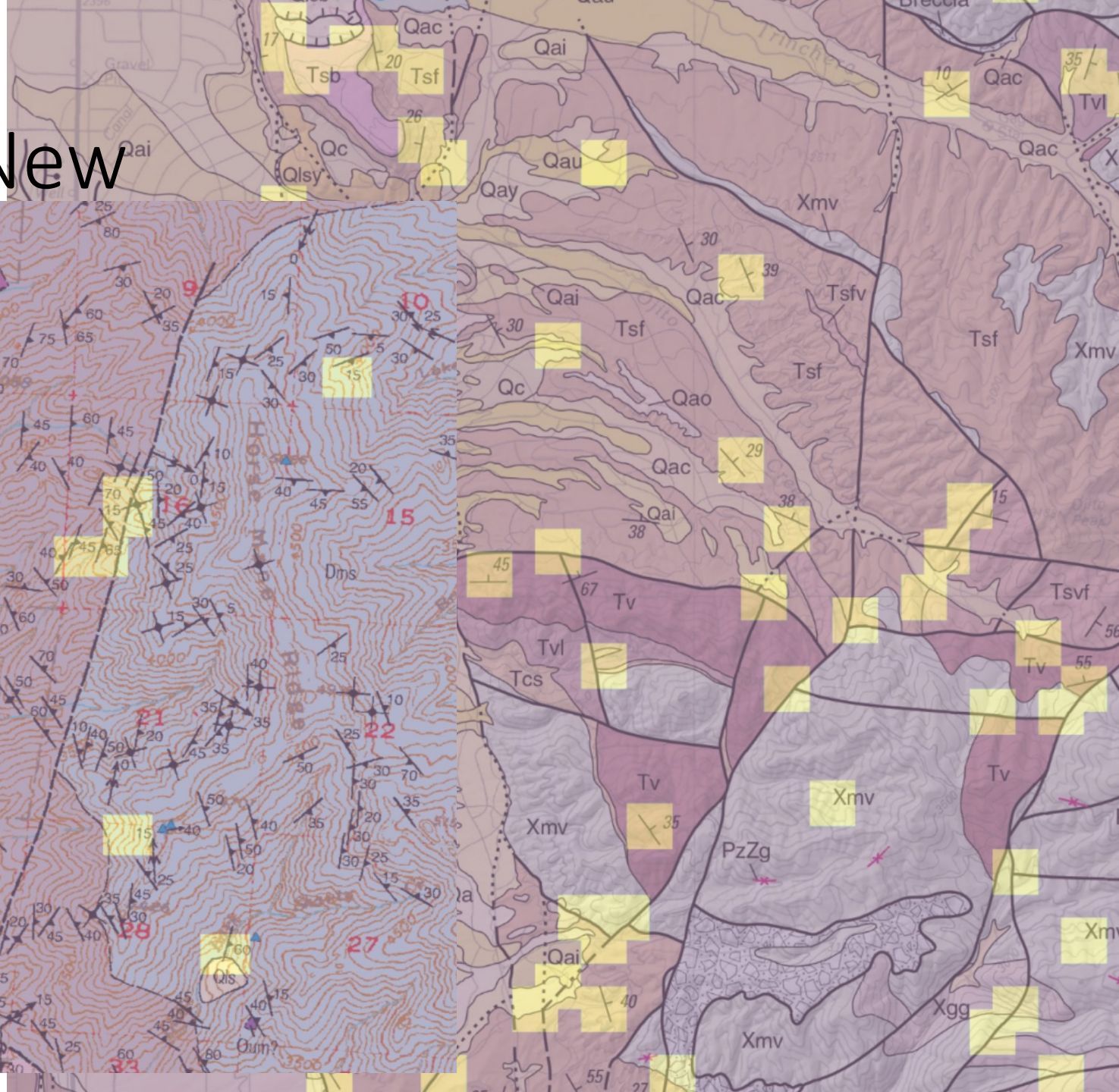


Sample Outputs - Good



Sample Outputs - New





Sample Outputs - New