Presentation 3: Running Multi-Stain Model

STAT 390 | Project 1 | Fall 2025

put
CROSS
STATIN
STATS-390

Multi-Stain Model Overview

```
          H&E Patches         SOX Patches         Melan Patches
                |                  |                   |
     [Pretrained CNN]     [Pretrained CNN]      [Pretrained CNN]
                |                  |                   |
[Patch-Level Attention] [Patch-Level Attention] [Patch-Level Attention]
                |                  |                   |
[Low-Level Slice Emb.] [Low-Level Slice Emb.]   [Low-Level Slice Emb.]
                |                  |                   |
[Slice-Level Attention] [Slice-Level Attention] [Slice-Level Attention]
                |                  |                   |
[Low-Level Stain Emb.] [Low-Level Stain Emb.]   [Low-Level Stain Emb.]
                |                  |                   |
[Stain-Level Attention] [Stain-Level Attention] [Stain-Level Attention]
                \                  |                  /
                 \                 |                 /
                  \________________|________________/
                                   |
                       [Low-Level Case Emb.]
                                   |
                           Classification
```

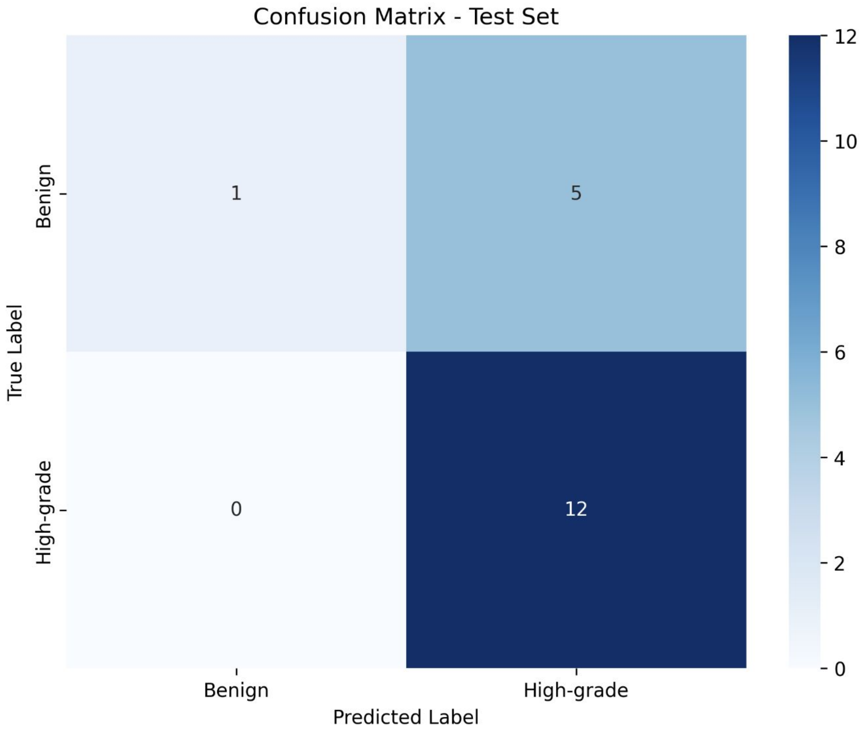
1. Initial Results

- Confusion matrix →

- Recall (high grade) = 12/12 = 1.0
- Recall (benign) = 1/6 ≈ 0.167
- Accuracy = 13/18 ≈ 0.722

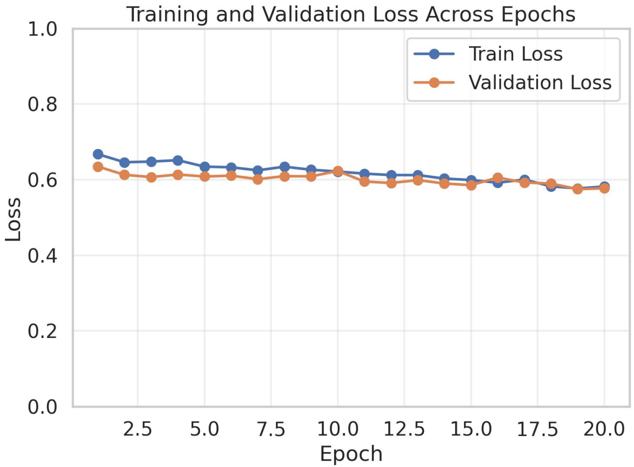
- Label distribution per split:

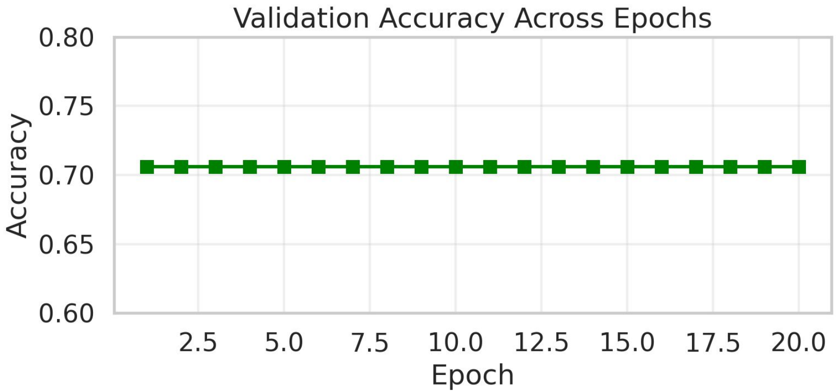
label	0	1
split		
test	6	12
train	17	34
val	5	12



1. Initial Results

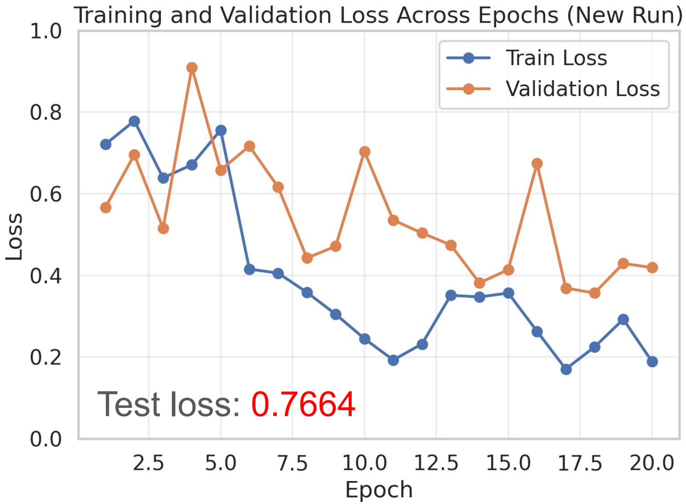
- Initial model showed strong bias toward predicting label = 1 (High-grade)
  - Misclassified almost all benign samples as high-grade
  - Caused by improper gradient flow
  - Learning rate also too low?

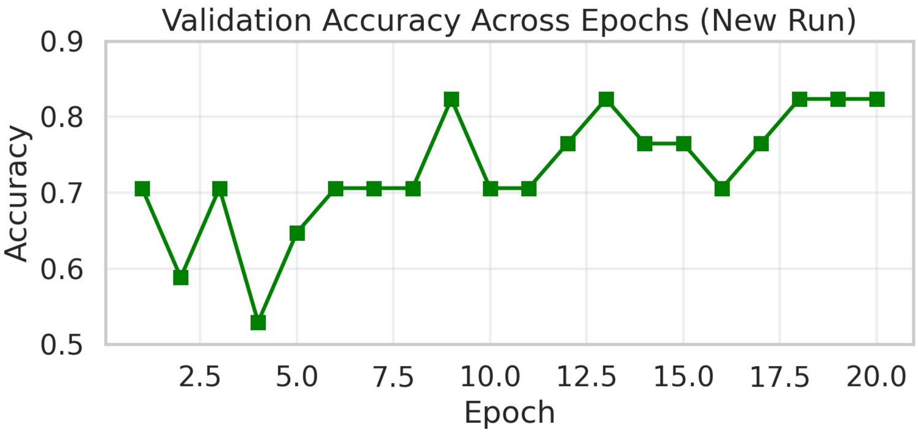




2. Updated Results

- The following fixes are made
  - Fixed gradient flow across all three layers (bug)
  - Increased learning rate from 1e-4 to 2e-4
  - Also 20 epochs





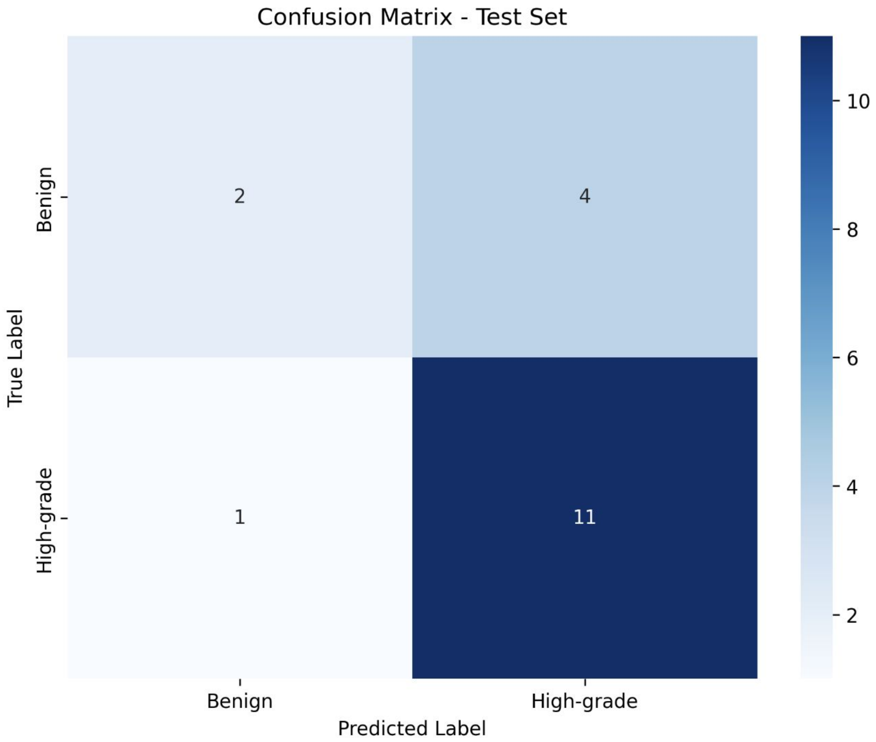
2. Updated Results

- Confusion matrix →

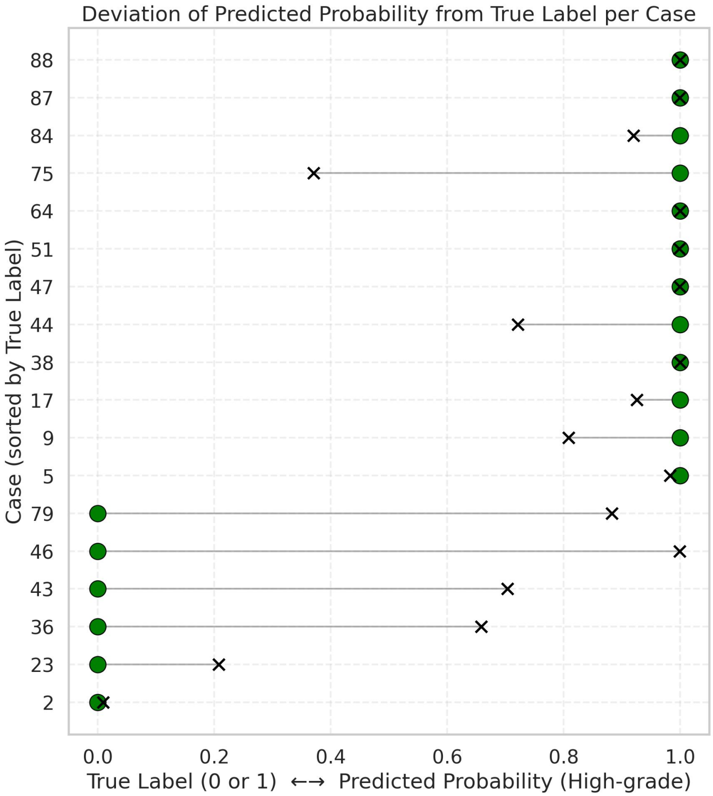
- Recall (high grade) = 11/12 ≈ 0.917
- Recall (benign) = 2/6 ≈ 0.333
- Accuracy = 13/18 ≈ 0.722

- Label distribution per split:

label	0	1
split		
test	6	12
train	17	34
val	5	12



2. Updated Results



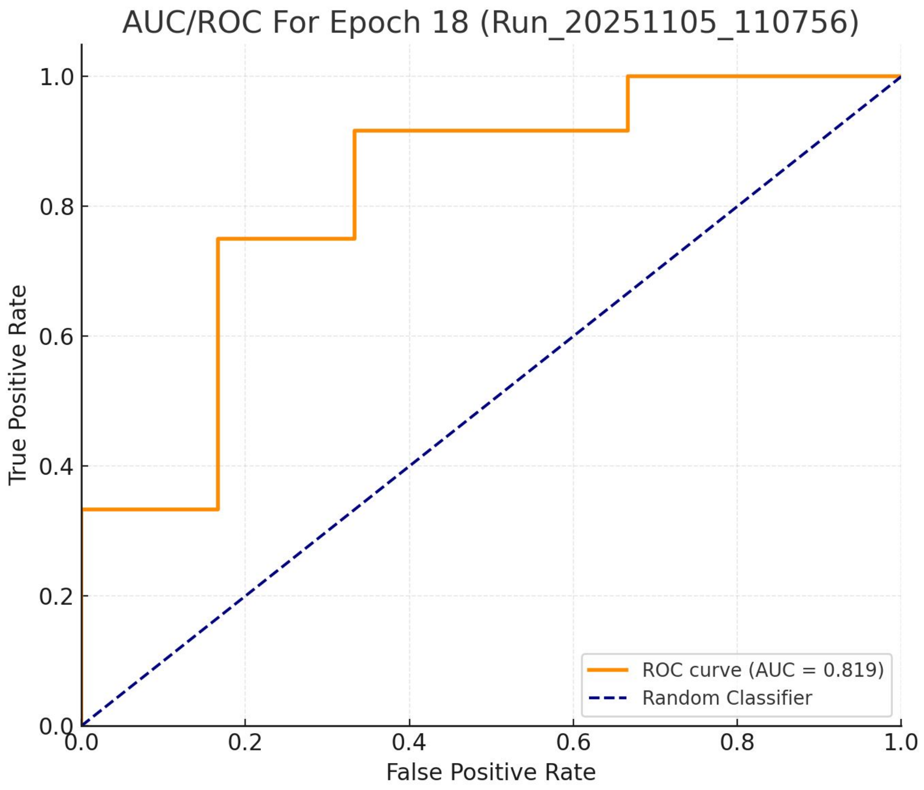
Green Dots: true label

X: predicted probability to be positive

Takeaway: most predictions tend to skew to the positive end.

Threshold tuning possible?

2. Updated Results



AUC = 0.819

- Solid performance!

2. Hyperparameter Tuning: Learning Rate

- **Learning rates tested:** 1e-5, 2e-5, 5e-5, 1e-4, 2e -4
- **Methodology:**
  - (1) GPT analyzed baseline val loss/acc
  - (2) Begin with wide grid search → finer grid tuning is running tonight

Learning Rate	Recall	Precision	Accuracy
1e-5	0.917	0.647	0.611
2e-5	0.917	0.647	0.611
5e-5	0.917	0.647	0.611
1e-4	0.917	0.647	0.611
2e-4	0.917	0.733	0.722

Takeaway: Keep trying larger learning rates (necessary with 20 epochs instead of 5)

2. Hyperparameter Tuning: Learning Rate

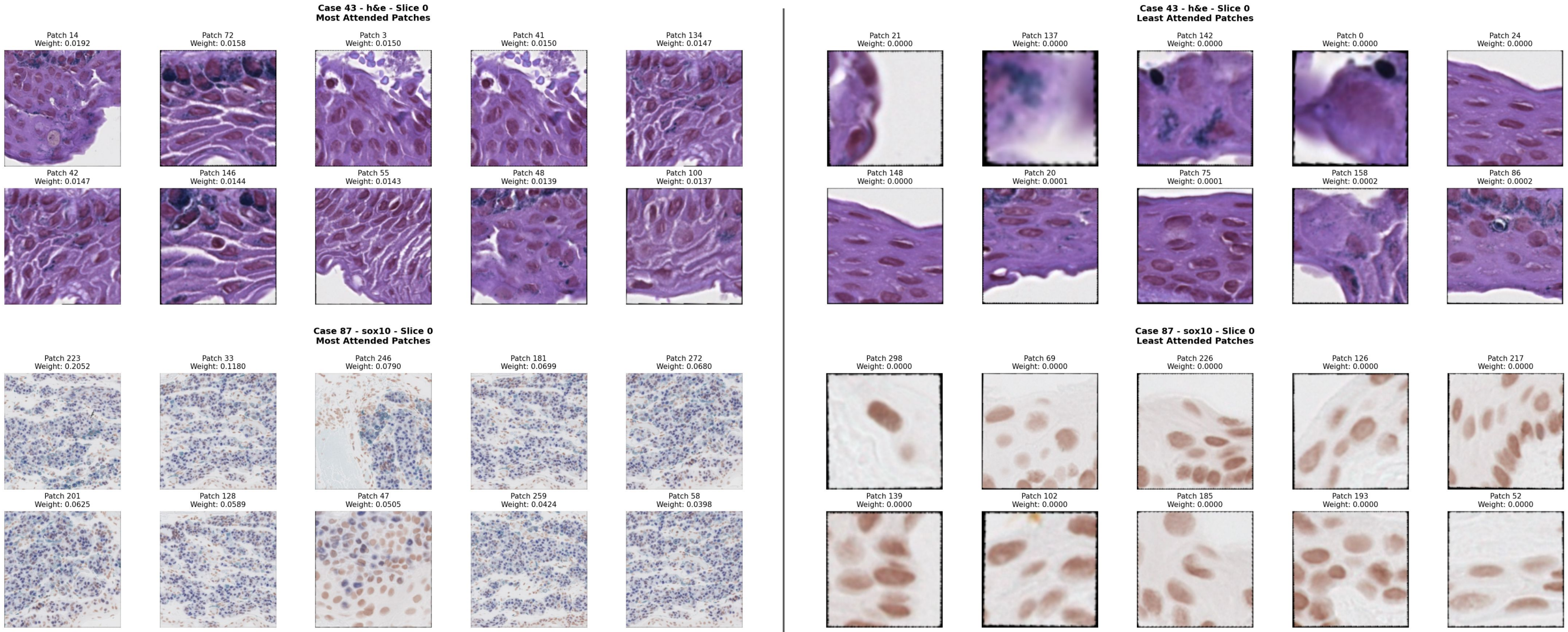
	Fixed LR	Scheduler LR
Pros	• Stable when properly tuned	• Reduces overfitting • Adapts throughout training process
Cons	• Not as suitable for multiple epochs & large datasets	• Longer training times
Next Steps	LRs 1.8e-4, 2.2e-4, 2.5e-4, 3e-4, 5e-4 (code running right now)	Link to code chunk

3. Attention Analysis: Patch-Level Results

- Attention mechanism is definitely working
- Patch level (when working really well):

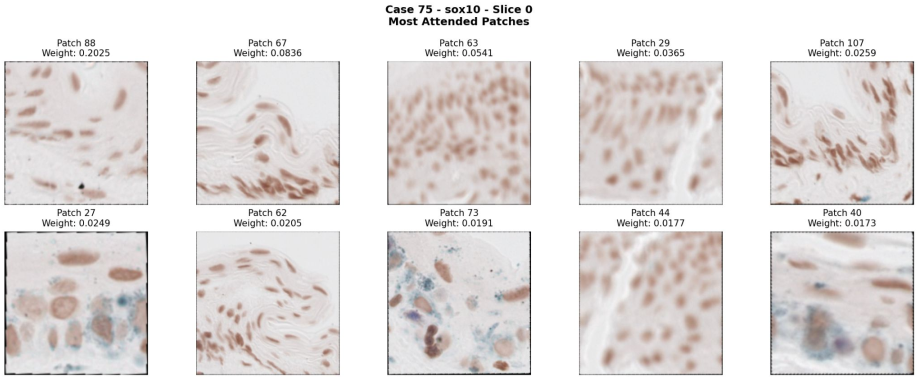
Case 43 (Up): Predicted = 1, Actual = 0

Case 87 (Down): Predicted = 1, Actual = 1

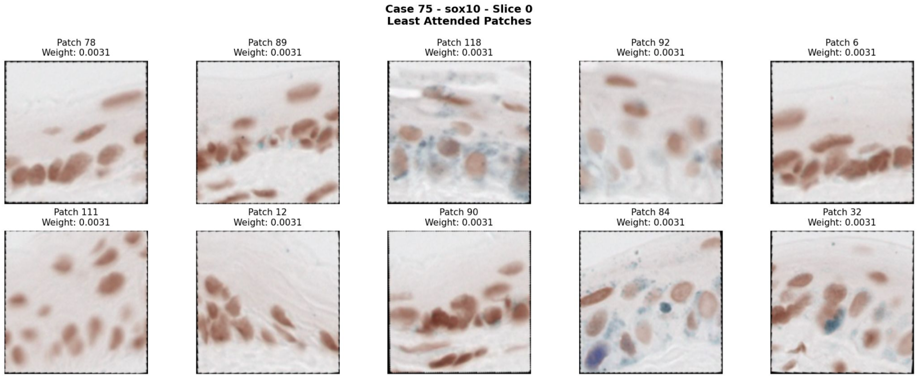


3. Attention Analysis: Slice-Level Results

- Patch level (when not that contrastive?):



Predicted = 0, Actual = 1



3. Attention Analysis: Slice-Level Results

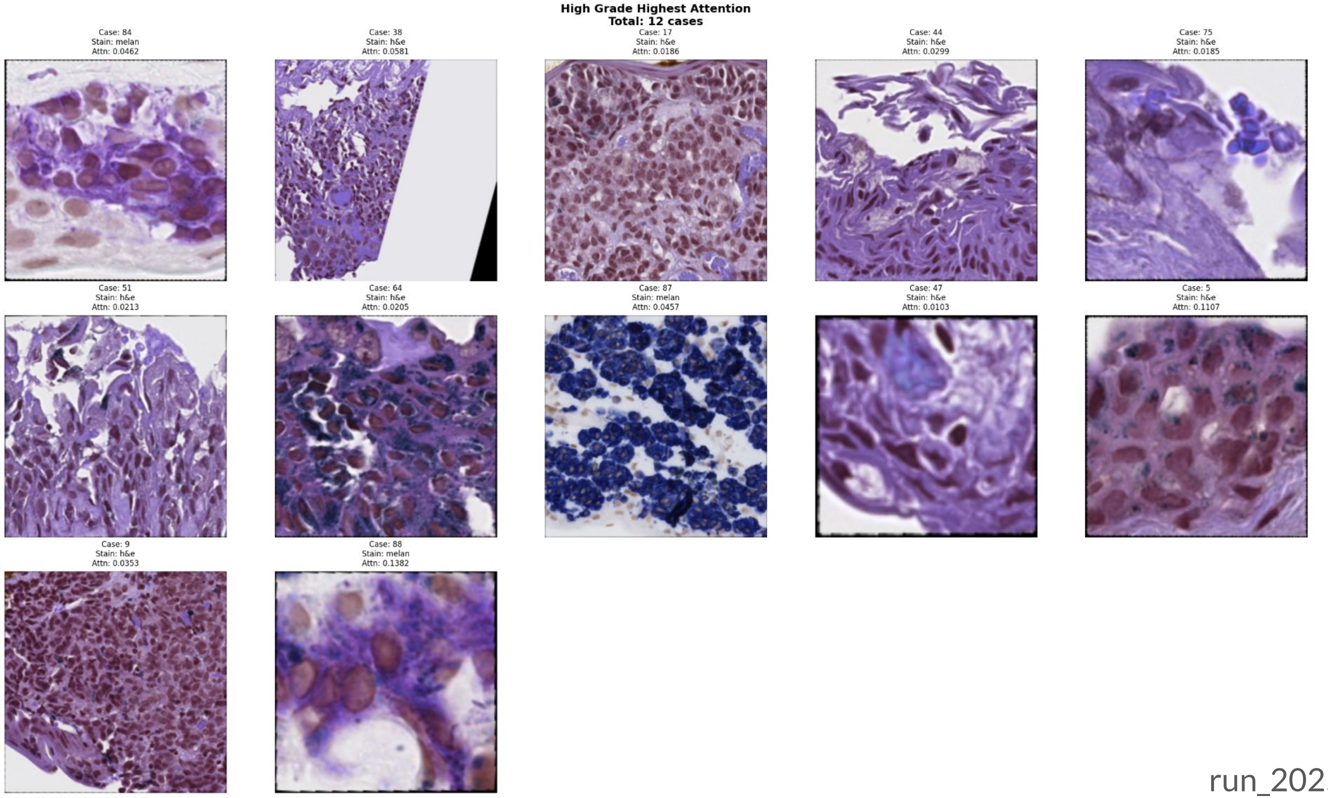
- **h&e stain** had highest attention for **15 cases**
- **melan stain** had highest attention for **3 cases**
- **sox10 stain** had highest attention for **0 cases**

From attention_summary.txt, run_20251105_110756

3. Attention Analysis: Visualizing Combinations

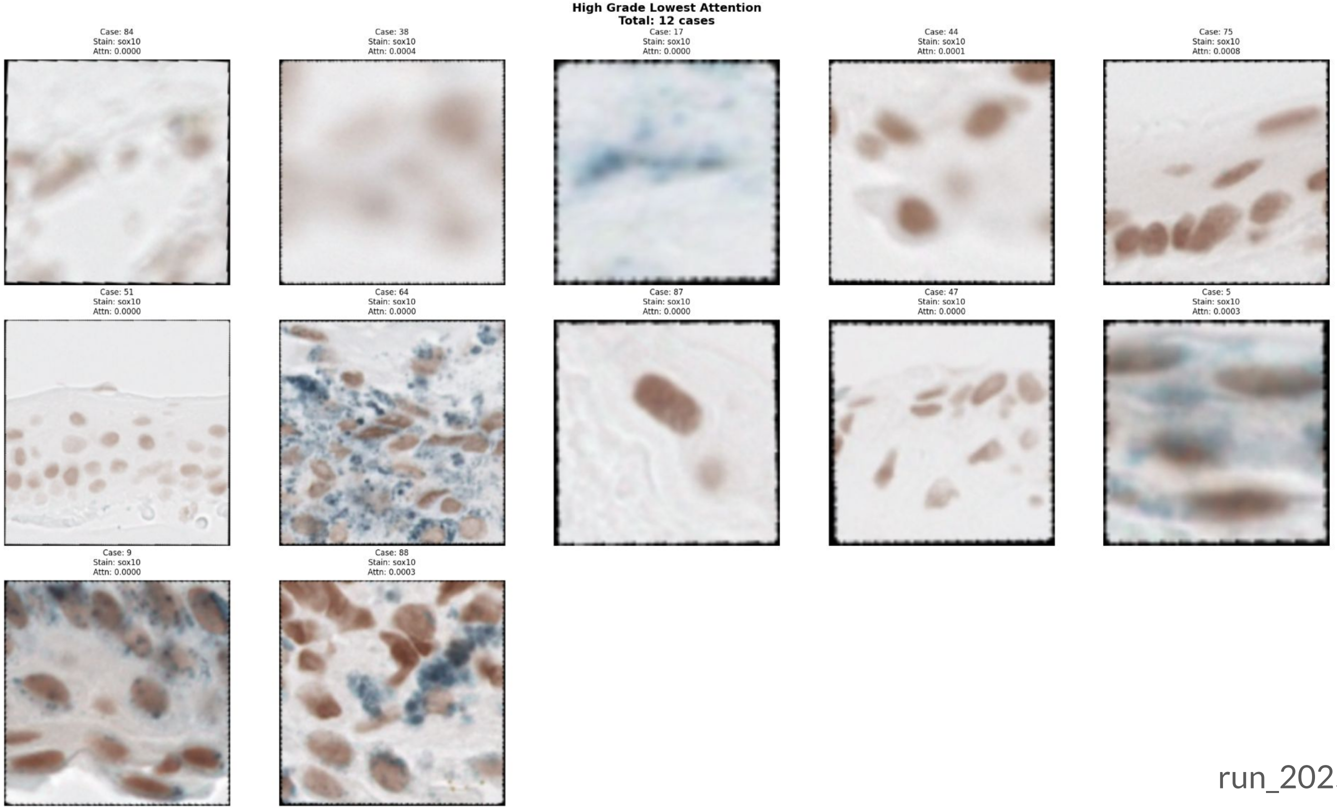
- **Goal:** Visualize patches of each combination–high grade, high attention; low grade, high attention; high grade, low attention; low grade, low attention
- **Approach [link to code chunk]:**
  For each case (high grade, benign):
    Highest attention patch:
    - Find highest attention stain (from case_weights)
    - Find highest attention slice within that stain
    - Visualize highest attention patch within that slice
    Lowest attention patch:
    - Find lowest attention stain (from case_weights)
    - Find lowest attention slice within that stain
    - Visualize lowest attention patch within that slice

3. Attention Analysis: High Grade, High Attention



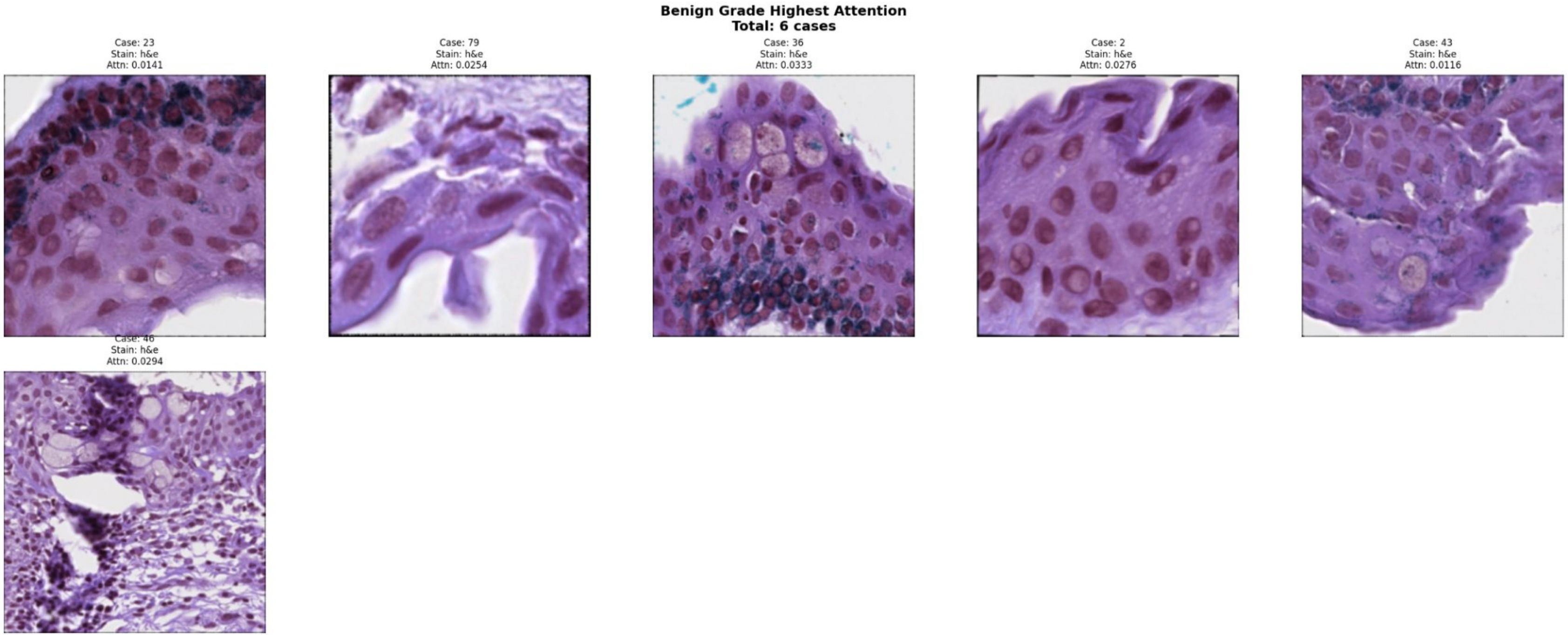
run_20251105_110756

3. Attention Analysis: High Grade, Low Attention

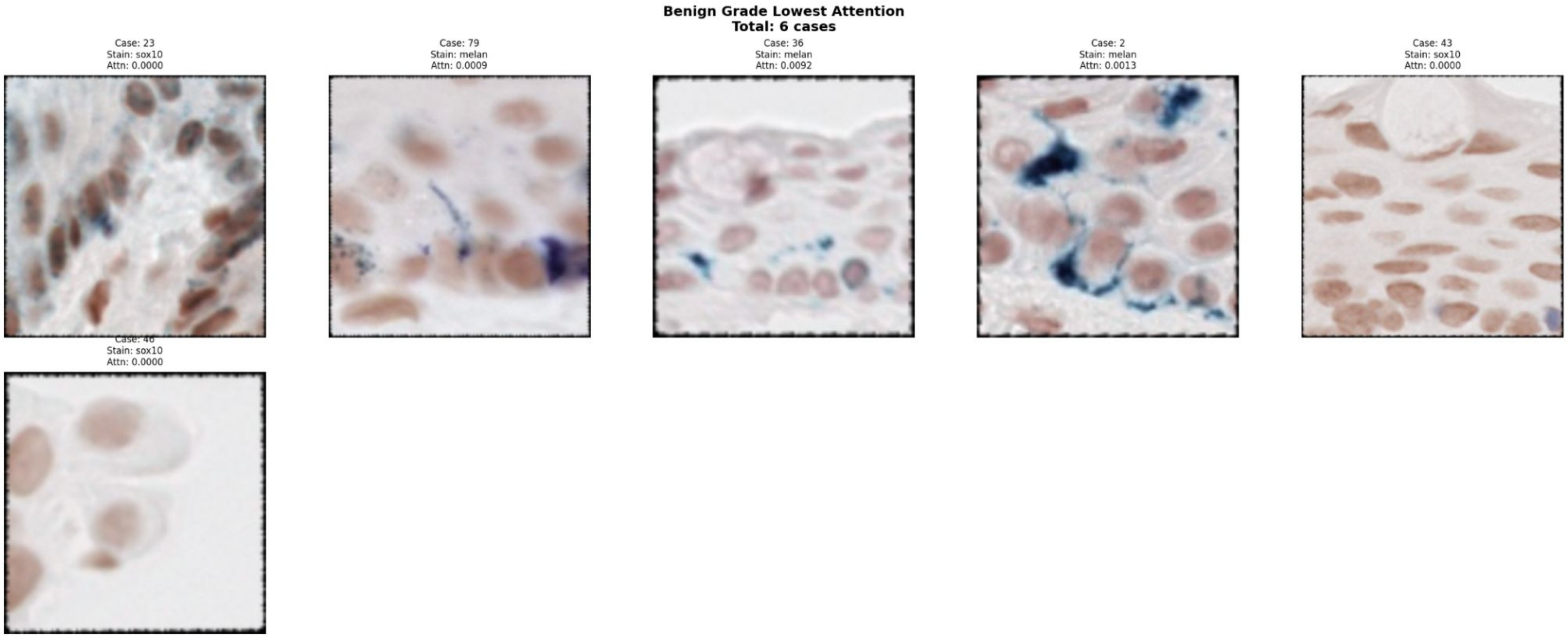


run_20251105_110756

3. Attention Analysis: Benign, High Attention



3. Attention Analysis: Benign, Low Attention



run_20251105_110756

4. Color Experimentation - HED

- HED color space has worse performance than RGB, but also without color normalization
- very close performance for high-grade on slice level

Patch Level | Slice Level | Case Level

RGB, 5 epoch, normalization

Patch Level

	precision	recall	f1-score	support
Benign	0.31	0.42	0.35	1706
High-grade CMIL	0.82	0.75	0.78	6287
accuracy			0.67	7993
macro avg	0.57	0.58	0.57	7993
weighted avg	0.71	0.67	0.69	7993

Slice Level

precision	recall	f1-score	support
0.60	0.69	0.64	13
0.94	0.91	0.92	66
		0.87	79
0.77	0.80	0.78	79
0.88	0.87	0.88	79

Case Level

precision	recall	f1-score	support
0.67	0.67	0.67	6
0.83	0.83	0.83	12
		0.78	18
0.75	0.75	0.75	18
0.78	0.78	0.78	18

Hematoxylin-Eosin-DAB, 5 epoch, no normalization

Patch Level

	precision	recall	f1-score	support
Benign	0.26	0.71	0.38	1706
High-grade CMIL	0.85	0.46	0.59	6287
accuracy			0.51	7993
macro avg	0.56	0.58	0.49	7993
weighted avg	0.73	0.51	0.55	7993

Slice Level

precision	recall	f1-score	support
0.53	0.62	0.57	13
0.92	0.89	0.91	66
		0.85	79
0.73	0.75	0.74	79
0.86	0.85	0.85	79

Case Level

precision	recall	f1-score	support
0.43	0.50	0.46	6
0.73	0.67	0.70	12
		0.61	18
0.58	0.58	0.58	18
0.63	0.61	0.62	18

4. Color Experimentation - Code Implementation

- **Goal:** try out different color space with new implementation. HSV, HED, LAB
- [link to code chunk]
- **Next Steps:**
  - get normalization parameters for each color space using a sample of training patches from each stain (need to migrate code into Quest)
  - Visualize transformed images in attention analysis

```python
def compute_hed_mean_std(dataset, sample_bags=300, sample_patches=15):
    """
    Compute mean and std of HED channels from a subset of the training dataset.
    dataset: SliceMILDataset with RGB2HED transform applied
    sample_bags: how many bags to sample
    sample_patches: how many patches to take from each bag
    """
    all_pixels = []

    # Randomly choose some bags from dataset
    sampled_indices = random.sample(range(len(dataset)), min(sample_bags, len(dataset)))

    for idx in tqdm(sampled_indices, desc="Sampling HED patches"):
        bag_imgs, _ = dataset[idx]  # (num_patches, 3, H, W)
        num_patches = bag_imgs.size(0)

        # Randomly pick up to sample_patches from each bag
        chosen = random.sample(range(num_patches), min(sample_patches, num_patches))
        sampled_imgs = bag_imgs[chosen]  # (M, 3, H, W)

        # Flatten and append (reshape to N×3)
        all_pixels.append(sampled_imgs.permute(0, 2, 3, 1).reshape(-1, 3))

    all_pixels = torch.cat(all_pixels, dim=0)  # shape: (total_pixels, 3)

    # Compute channel-wise mean and std
    mean = all_pixels.mean(dim=0)
    std = all_pixels.std(dim=0)

    return mean, std
```

```python
hed_mean, hed_std = compute_hed_mean_std(train_ds, sample_bags=100, sample_patches=10)
print("HED Mean:", hed_mean)
print("HED Std:", hed_std)
```

5. Next Steps

1) Investigate ways to make the gradient descent smoother and more stable
2) Investigate how slice-level attention varies, especially for slices split into sub-slices

   → Can we remove slice level for some / all cases?
3) Investigate transformations / large grey areas
4) More complex attention modules; add more fully-connected layers at the end classification

References

GenAI tools