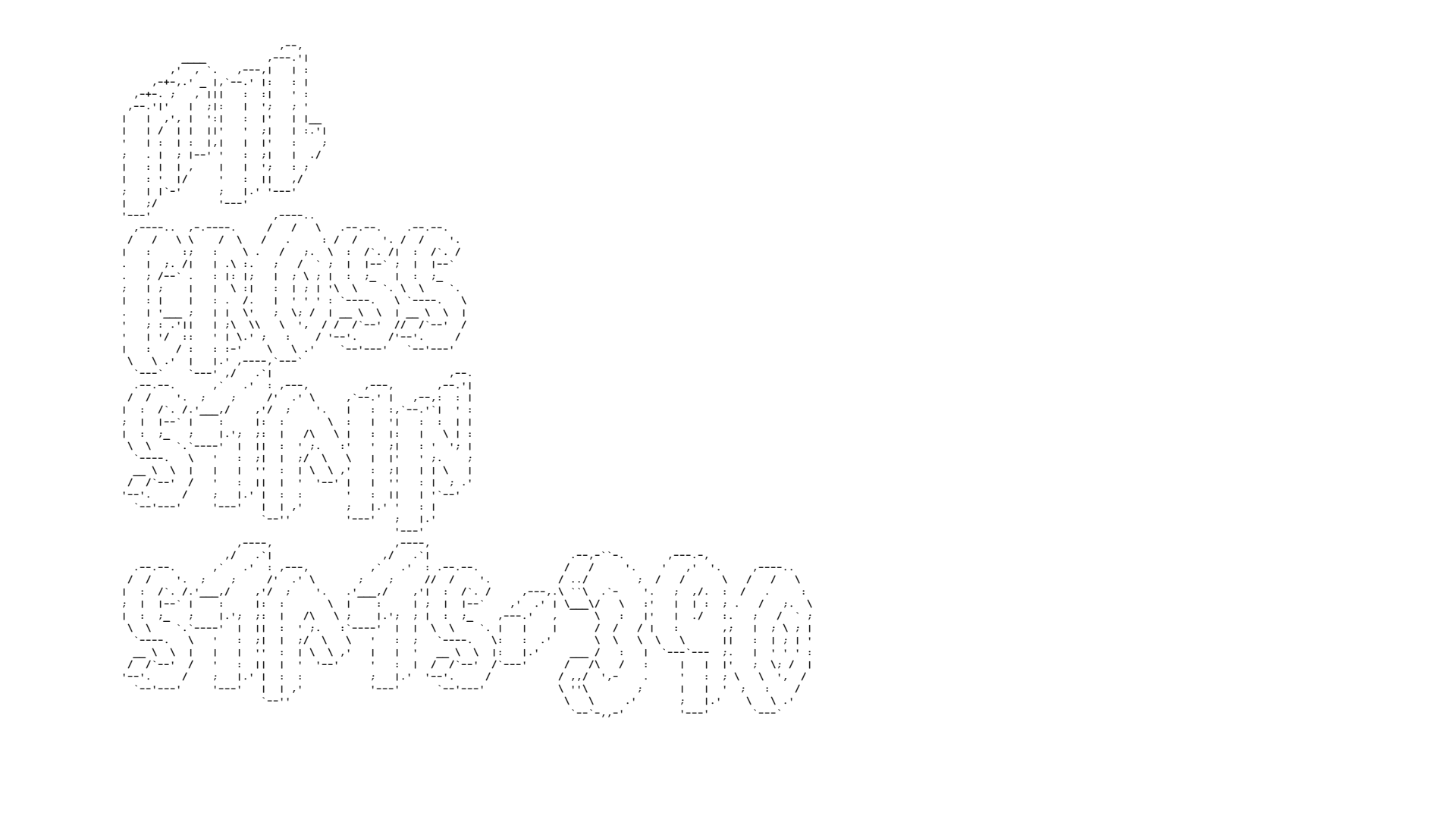
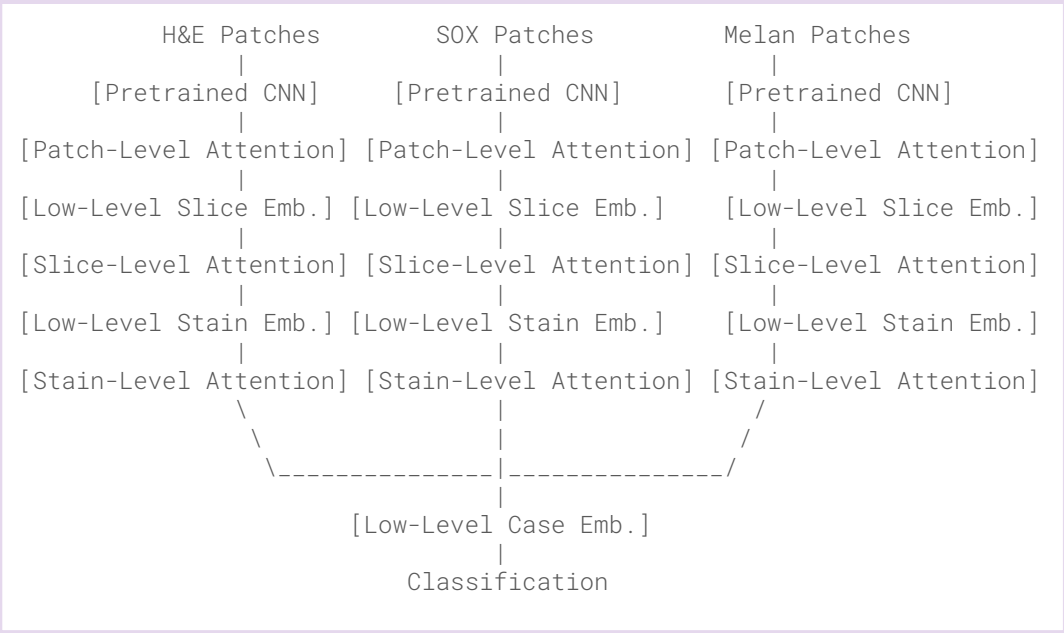
Presentation 2: Revision of Spring Code + Coding Multi-Stain

STAT 390 | Project 1 | Fall 2025



Multi-Stain Model Overview



Point of discussion on whether or not to include slice-level attention; current code does

Multi-Stain Model Example

```
 H&E:
  Slice 1: [50 patches] → Patch Attention → Slice Embedding
  Slice 2: [50 patches] → Patch Attention → Slice Embedding
  → Stain Attention → H&E Stain Embedding

SOX10:
  Slice 1: [30 patches] → Patch Attention → Slice Embedding
  → Stain Attention → SOX10 Stain Embedding

Melan:
  Slice 1: [40 patches] → Patch Attention → Slice Embedding
  → Stain Attention → Melan Stain Embedding

[H&E Emb, SOX10 Emb, Melan Emb] → Case Attention → Case
Embedding → Classification
```

1. Revising Spring Code: Data Leakage Fix

- **Spring data leakage**
  - Split data into train / validation / test at the **slice level** rather than the case level (data leakage)
- **Updated split**
  - Corrected to split data into train / validation / test at the **case level**
    - Start from the original data structure `{0: [(case_i, slice_j), …], 1:...}`
    - Extract the case mappings `{case_i: label_i, …}`
    - Leverage train_test_split on the case level
    - Return:
      - three lists (train, test, val): `[(case_a, slice_1), (case_b, slice_2), ...]`
      - …and case mappings to labels.
  - (Link to code chunk)

1. Revising Spring Code: Other Fixes

1) **Incomplete RegEx code** due to misnamed patches; identified format of misnamed patches and included these in matching

   → Result: +3808 patches matched, +28 slices across train / validation / test

   (github code chunk, colab code chunk)

2) Redundant if-statements / checks

   (Various code chunks; very minor changes)

1. Revising Spring Code: Other Fixes

1) Small fix of saving path to drive instead of local path
2) Automate checkpoint loading: Adding load_lastest checkpoint() function, and start_epoch variable in train_model

```python
# check if we need to load from checkpoint before training
checkpoint_dir = "/content/drive/MyDrive/Checkpoints"
checkpoint_pattern = re.compile(r'epoch(\d+)\.pth')
start_epoch = 0
checkpoint_path = None
checkpoint_files = [f for f in os.listdir(checkpoint_dir) if f.endswith('.pth')]

def load_latest_checkpoint(checkpoint_dir, model, optimizer, device):
    # Find all files in the directory
    checkpoint_files = [f for f in os.listdir(checkpoint_dir) if checkpoint_pattern.search(f)]
    if len(checkpoint_files)>=1:
        # Sort by extracted epoch number
        checkpoint_files.sort(
            key=lambda x: int(checkpoint_pattern.search(x).group(1)) if checkpoint_pattern.search(x) else -1)
        # Get the path of the latest checkpoint (last one in the sorted list)
        checkpoint_path = os.path.join(checkpoint_dir, checkpoint_files[-1])
        print(f"Latest checkpoint in directory is **{checkpoint_path}**")

        checkpoint = torch.load(checkpoint_path, map_location=device)
        model.load_state_dict(checkpoint['model_state_dict'])
        optimizer.load_state_dict(checkpoint['optimizer_state_dict'])
        start_epoch = checkpoint['epoch']
        return model, optimizer, start_epoch
    return model, optimizer, 0

model, optimizer, start_epoch = load_latest_checkpoint(checkpoint_dir, model, optimizer, device)
```

```python
def train_model(model, optimizer, criterion, train_loader, val_loader, arch, epochs=5, start_epoch=0):
    model.train()

    for epoch in range(start_epoch, epochs):
        running_loss = 0.0
```

1. Results From Spring Code

- With 5 epochs, h&e stains only, DenseNet + Attention MIL

🔹 Train Split
Cases: 51
Slices: 197
Patches: 23912
Class distribution: {1: 109, 0: 88}

🔹 Validation Split
Cases: 17
Slices: 60
Patches: 7832
Class distribution: {1: 44, 0: 16}

🔹 Test Split
Cases: 18
Slices: 79
Patches: 7993
Class distribution: {1: 66, 0: 13}

=== Slice-Level Classification Report ===

	precision	recall	f1-score	support
Benign	0.60	0.69	0.64	13
High-grade CMIL	0.94	0.91	0.92	66
accuracy			0.87	79
macro avg	0.77	0.80	0.78	79
weighted avg	0.88	0.87	0.88	79

=== Patch-Level Classification Report (weak labels) ===

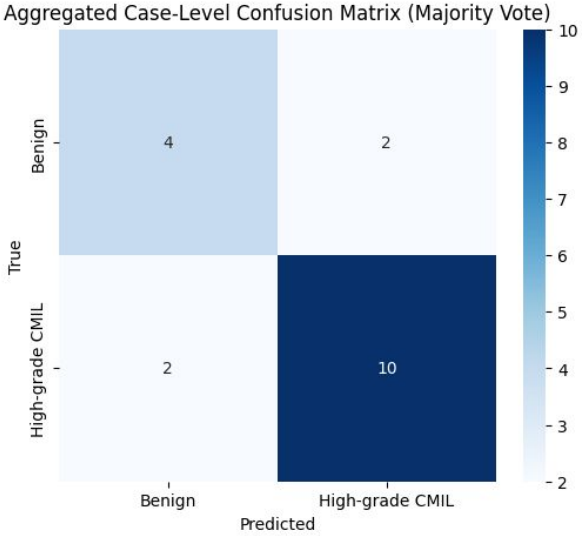
	precision	recall	f1-score	support
Benign	0.31	0.42	0.35	1706
High-grade CMIL	0.82	0.75	0.78	6287
accuracy			0.67	7993
macro avg	0.57	0.58	0.57	7993
weighted avg	0.71	0.67	0.69	7993

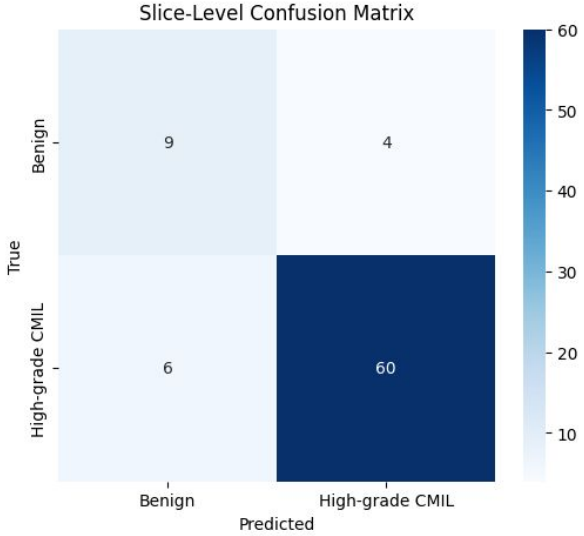
=== Aggregated Case-Level Classification Report ===

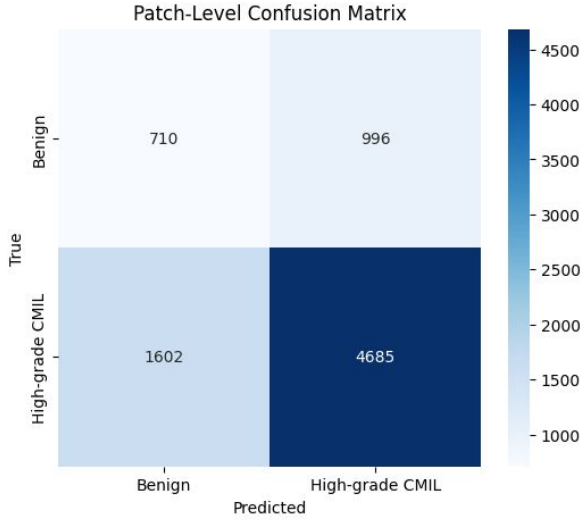
	precision	recall	f1-score	support
Benign	0.67	0.67	0.67	6
High-grade CMIL	0.83	0.83	0.83	12
accuracy			0.78	18
macro avg	0.75	0.75	0.75	18
weighted avg	0.78	0.78	0.78	18

1. Results From Spring Code (Cont.)

- Confusion Matrices



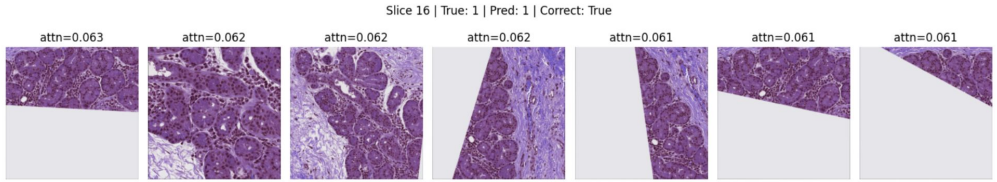




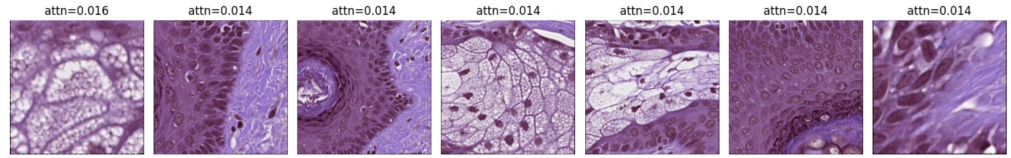
1. Results From Spring Code (Cont.)

- Slice-level Attention Scores (link to code chunk)

```
✓ Slice #16: True label = 1, Predicted = 1, Correct = True
Top patches with highest attention:
  - case_104_match_12_h&e_patch8.png: attention = 0.0626
  - case_104_match_12_h&e_patch16.png: attention = 0.0620
  - case_104_match_12_h&e_patch0.png: attention = 0.0619
  - case_104_match_12_h&e_patch15.png: attention = 0.0619
  - case_104_match_12_h&e_patch14.png: attention = 0.0615
  - case_104_match_12_h&e_patch9.png: attention = 0.0607
  - case_104_match_12_h&e_patch13.png: attention = 0.0605
```




```
✓ Slice #13: True label = 1, Predicted = 0, Correct = False
Top patches with highest attention:
  - case_104_unmatched_1_h&e_patch30.png: attention = 0.0163
  - case_104_unmatched_1_h&e_patch59.png: attention = 0.0143
  - case_104_unmatched_1_h&e_patch54.png: attention = 0.0142
  - case_104_unmatched_1_h&e_patch32.png: attention = 0.0142
  - case_104_unmatched_1_h&e_patch29.png: attention = 0.0138
  - case_104_unmatched_1_h&e_patch119.png: attention = 0.0138
  - case_104_unmatched_1_h&e_patch148.png: attention = 0.0138
```



2. Preparing for Cross-Stain MIL

- Data loader (link to code chunk)
  a) Each case is a batch (batch_size = 1)
  b) Each batch, returned by .__getitem__(idx) in the dataset object, is structured as:

```
{
    "case_id": case_id,
    "stain_slices": stain_slices,  # dict[stain] -> list[Tensor(P,C,H,W)]
    "label": label,
}
```

Sample:

```
Case ID: 82
      Label: 0
      h&e: 6 slices | 1091 patches total
        example slice shapes: [(255, 3, 224, 224), (138, 3, 224, 224), (143, 3, 224, 224)]
      melan: 1 slices | 320 patches total
        example slice shapes: [(320, 3, 224, 224)]
      sox10: 1 slices | 337 patches total
        example slice shapes: [(337, 3, 224, 224)]
```

2. Implementing Cross-Stain MIL

- Updated architecture: cross-stain, using new data loader
    a) Updated AttnMIL (github code chunk, colab code chunk)

    → Now uses attention at patch, slice, and stain levels

    b) Updated train_model (github code chunk, colab code chunk)

    c) Updated architecture implementation (github code chunk, colab code chunk)
- Next steps
    a) Introduce padding for stains with fewer slices

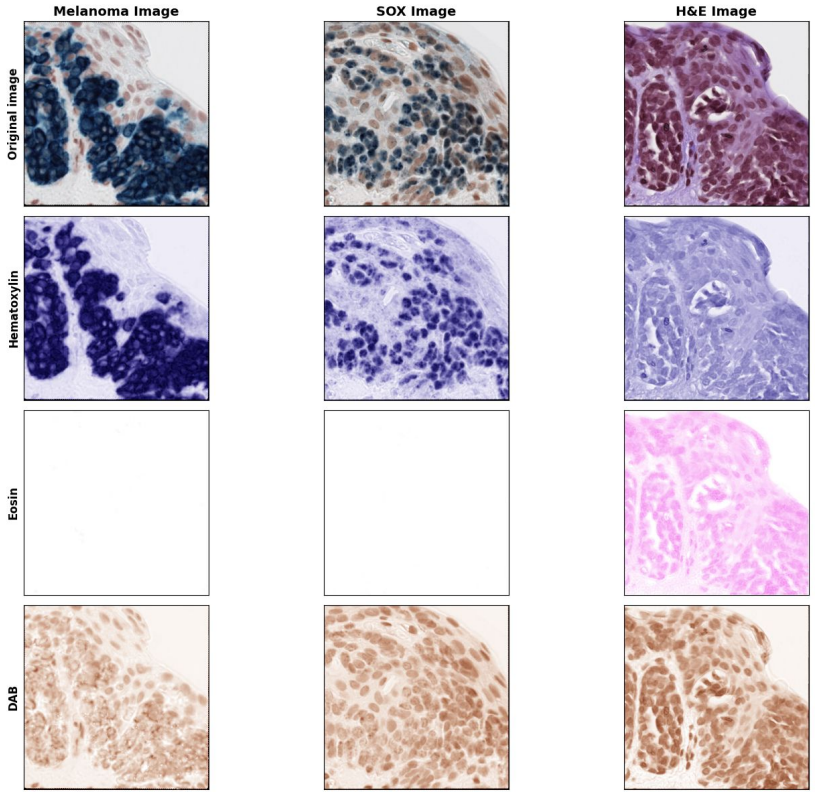
    b) Confirm if batch_size = 1 is ideal

    c) Add additional linear layers

3. H&E Color Deconvolution Research

- `Scikit-image.color` has a function `hedtorgb()`
- Channels:
  - Hematoxylin (H): Stains cell nuclei a blue/purple color
  - Eosin (E): Stains the cytoplasm a pink/red colors
  - DAB: location of target antigen
- IHC stains: H & DAB channels
- H & E stains: primarily H & E channels

- Idea Discussion:
  a. replace R-G-B with H-E-DAB, add 0 to E channels for IHC stains
  b. reduce to 2 dimensions: H-E for H&E and H-DAB for IHCs



https://scikit-image.org/docs/0.25.x/auto_examples/color_exposure/plot_ihc_color_separation.html

References

https://scikit-image.org/docs/0.25.x/auto_examples/color_exposure/plot_ihc_color_separation.html