Tymo’s progress

Week 2

Goals

OCR

Optical Character Recognition

OCR

Optical Character Recognition (OCR) is a foundational technology behind the conversion of typed, handwritten or printed text from images into machine-encoded text.

This could be used to make a new temporal feature: “Elapsed time” for every frame in the examination.

OCR

The elapsed time since the beginning of the examination is being shown here.

This could give us a better understanding of time during the examinations.

OCR

There are 2 timeframes, 1 is from when the FA frame was shot and the other one is (probably!) from when the ICGA frame was shot.

! This is not sure, needs to be checked with specialist !

OCR

This timeframe would be interesting to add as “metadata” to our frames, then we can make better use of the FA phases.

OCR

There seem to be 5 phases during an FA exam (link)
(i) 9-15 seconds : *Choroidal phase (AKA pre-arterial phase)*
The hyperfluorescence is present, a cilioretinal artery* if there is one, will fill in this phase

(ii) 1-3 seconds later : *Arterial phase*
Ateries are bright, but the veins remain dark

(iii) Arteriovenous phase
Laminar flow in the veins – The walls of the veins are bright while the center of the vein is still dark

(iv) 30 seconds : *Venous phase*
Complete filling of the veins

(v) 30 seconds - 10 minutes
Dye has recirculated, things that are going to leak or pool have already done so

OCR

Option A

Buckets:

- **Early** : *(0-25sec)*
- **Mid** : *(25-120sec)*
- **Late** : *(>120sec)*

Option B

Buckets:

- **Arterial** : *(<15sec)*
- **AV** : *(15-30 sec)*
- **Venous** *(30-120sec)*
- **Late** *(>120sec)*

! These are guidelines, there are a few factors that impact the time it takes for the dye to spread, e.g., age and cardiac condition.

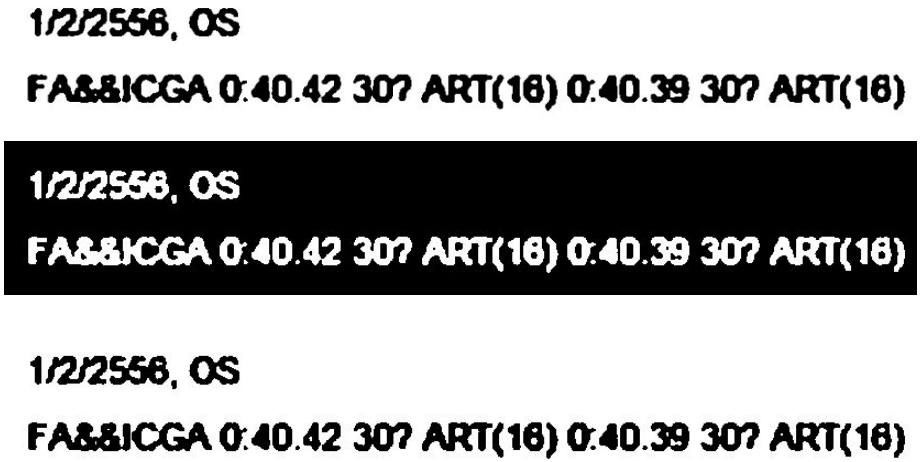
OCR

So all of this would make the use of OCR really interesting! These new elapsed time features could come in real handy.

First I got my hands on “EasyOCR” which did sound promising, especially the “Easy” part.

As a first step I pre processed the image, which means I cropped at and changed some of the image settings to gain a better view for the EasyOCR reader to do it’s work.

OCR



This was my favourite, inverted colours + scaling

OCR

At first, I used the EasyOCR python library to just see what it could extract from this image.
This resulte in the following:


```
[OCR raw] 1722556,05 FAEBICGA 0.40.42 307 ART(10) 0.40.39 307 ART(10)
```

It didn't manage to extract everything perfectly, but it got the times right!

The reason I'm also using the ICGA time now, is to use it as a fallback system for later

OCR

By implementing a Regex, I could extract the time easily from the raw output string:

```
pattern = r"\d{1,2}[:.]\d{2}\.\d{2}"
times = re.findall(pattern, all_text)
```

```
Elapsed times found: ['0:40.42', '0:40.39']
FA elapsed time: 0:40.42
ICGA elapsed time: 0:40.39
```

Now I had it working well for one specific FA image, it was then time to test it on a lot of other FA images.

OCR

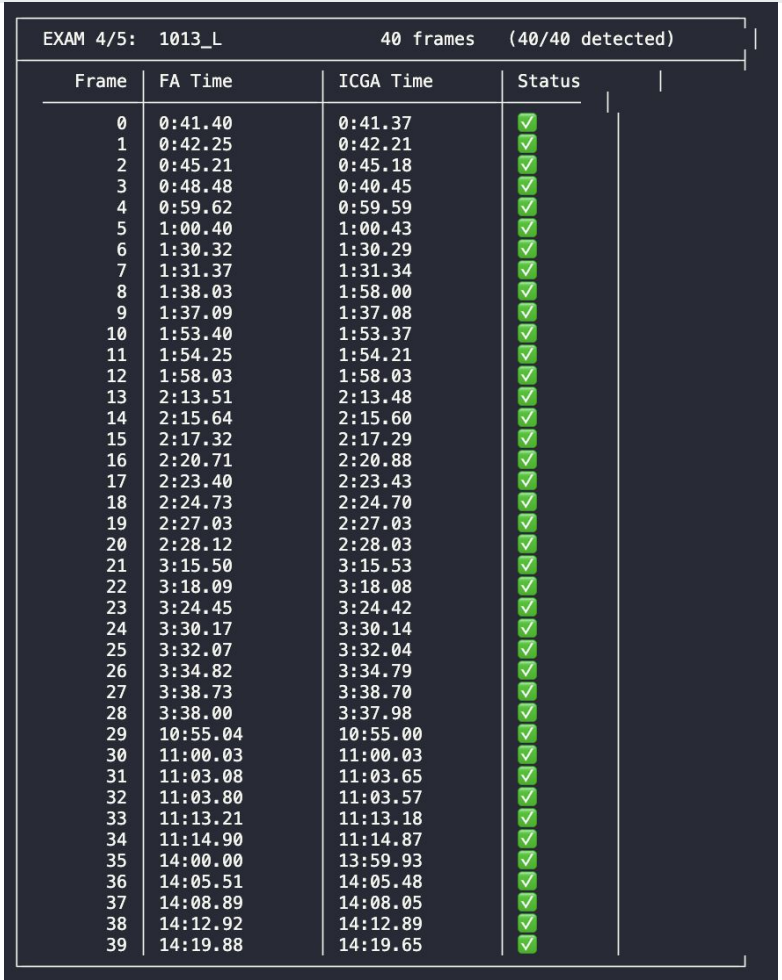
The final experiment resulted in the following results:

Now it was time to store this metadata in a .csv for every frame out of every exam. To later be able to really store it inside phases/buckets.

When there is no FA Time, or it is nog logical

n < n-1 or too big of a jump, it will use the ICGA time.

If the ICGA time is not valid either, it will interpolate between nearest valid neighbours



EXAM 4/5: 1013_L 40 frames (40/40 detected)

Frame	FA Time	ICGA Time	Status
0	0:41.40	0:41.37	✅
1	0:42.25	0:42.21	✅
2	0:45.21	0:45.18	✅
3	0:48.48	0:40.45	✅
4	0:59.62	0:59.59	✅
5	1:00.40	1:00.43	✅
6	1:30.32	1:30.29	✅
7	1:31.37	1:31.34	✅
8	1:38.03	1:58.00	✅
9	1:37.09	1:37.08	✅
10	1:53.40	1:53.37	✅
11	1:54.25	1:54.21	✅
12	1:58.03	1:58.03	✅
13	2:13.51	2:13.48	✅
14	2:15.64	2:15.60	✅
15	2:17.32	2:17.29	✅
16	2:20.71	2:20.88	✅
17	2:23.40	2:23.43	✅
18	2:24.73	2:24.70	✅
19	2:27.03	2:27.03	✅
20	2:28.12	2:28.03	✅
21	3:15.50	3:15.53	✅
22	3:18.09	3:18.08	✅
23	3:24.45	3:24.42	✅
24	3:30.17	3:30.14	✅
25	3:32.07	3:32.04	✅
26	3:34.82	3:34.79	✅
27	3:38.73	3:38.70	✅
28	3:38.00	3:37.98	✅
29	10:55.04	10:55.00	✅
30	11:00.03	11:00.03	✅
31	11:03.08	11:03.65	✅
32	11:03.80	11:03.57	✅
33	11:13.21	11:13.18	✅
34	11:14.90	11:14.87	✅
35	14:00.00	13:59.93	✅
36	14:05.51	14:05.48	✅
37	14:08.89	14:08.05	✅
38	14:12.92	14:12.89	✅
39	14:19.88	14:19.65	✅

OCR

Found out that there is also something else in the frame that could be of use: the **ART**

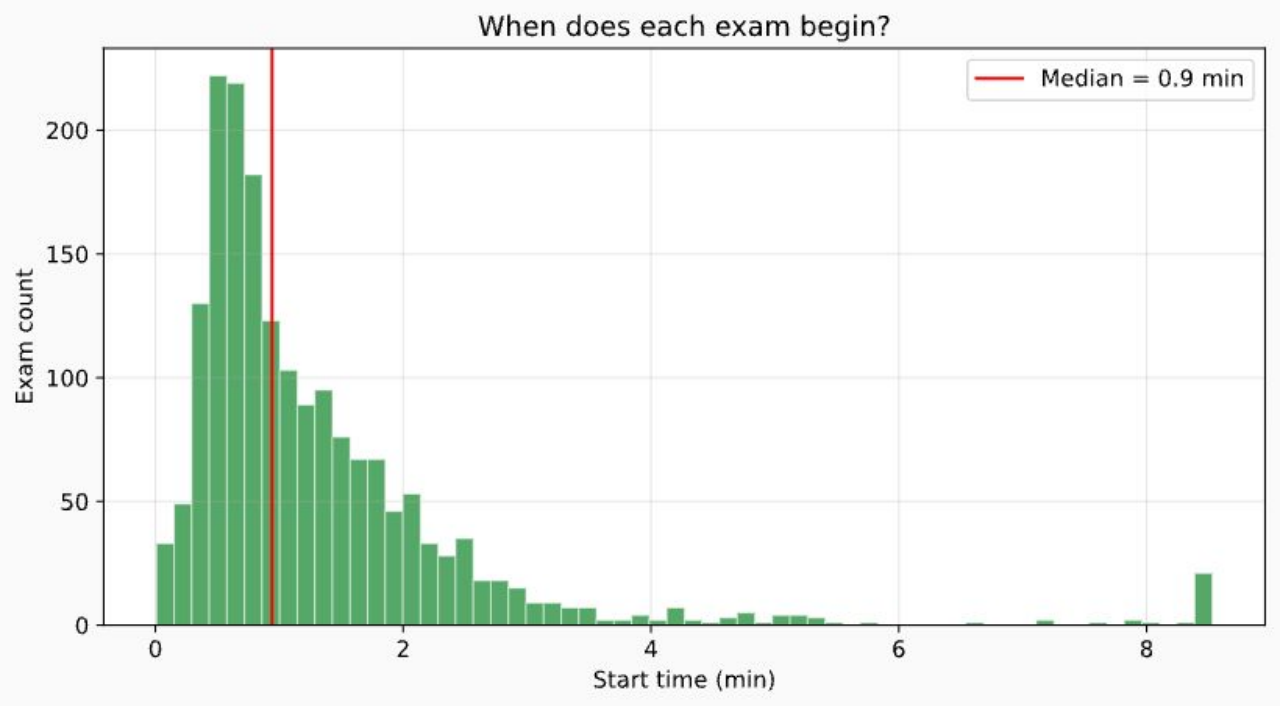
ART (Automatic Real-Time) averages multiple frames captured at the exact same retinal location.

So it could give an indication of the image quality, which could come in handy when picking images for the buckets

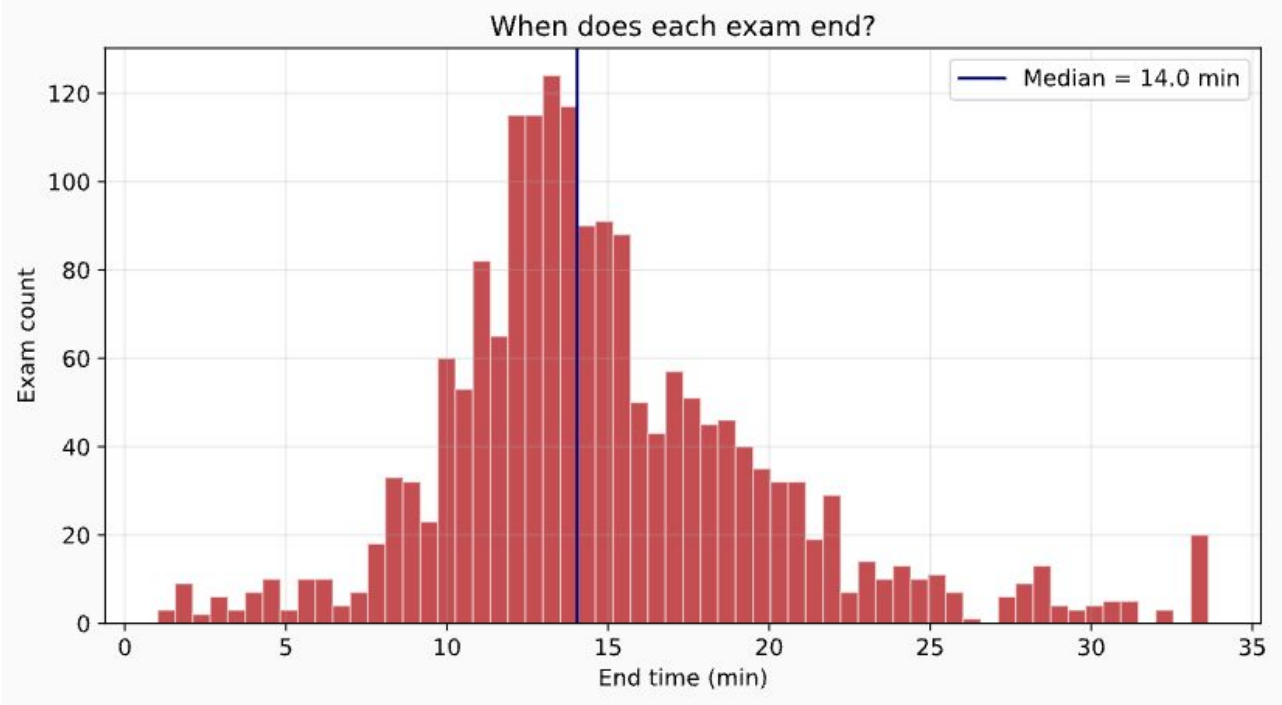
Most of the frames seem to have an ART of 16, but there are some cases where this is higher / lower.

OCR; Results

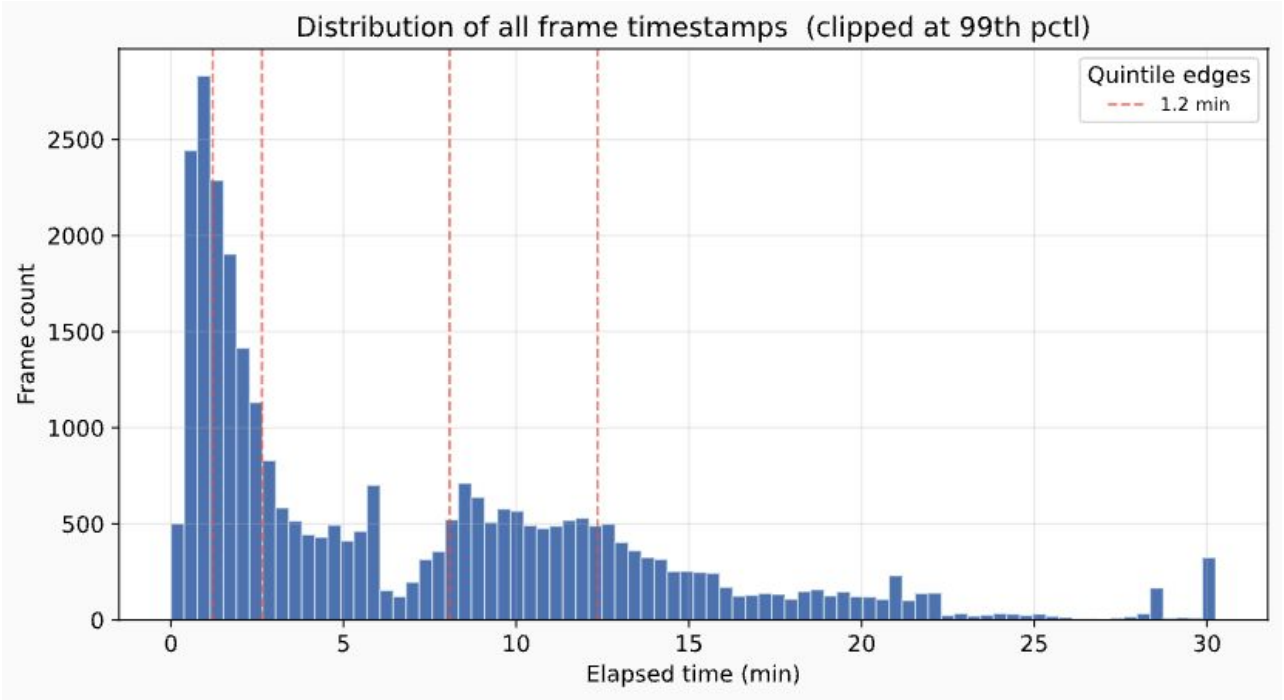
These are the results of ~1300 exams, since I more or less knew these had somewhat correct values, other ones had some big outliers so I didn't trust them enough to use them.



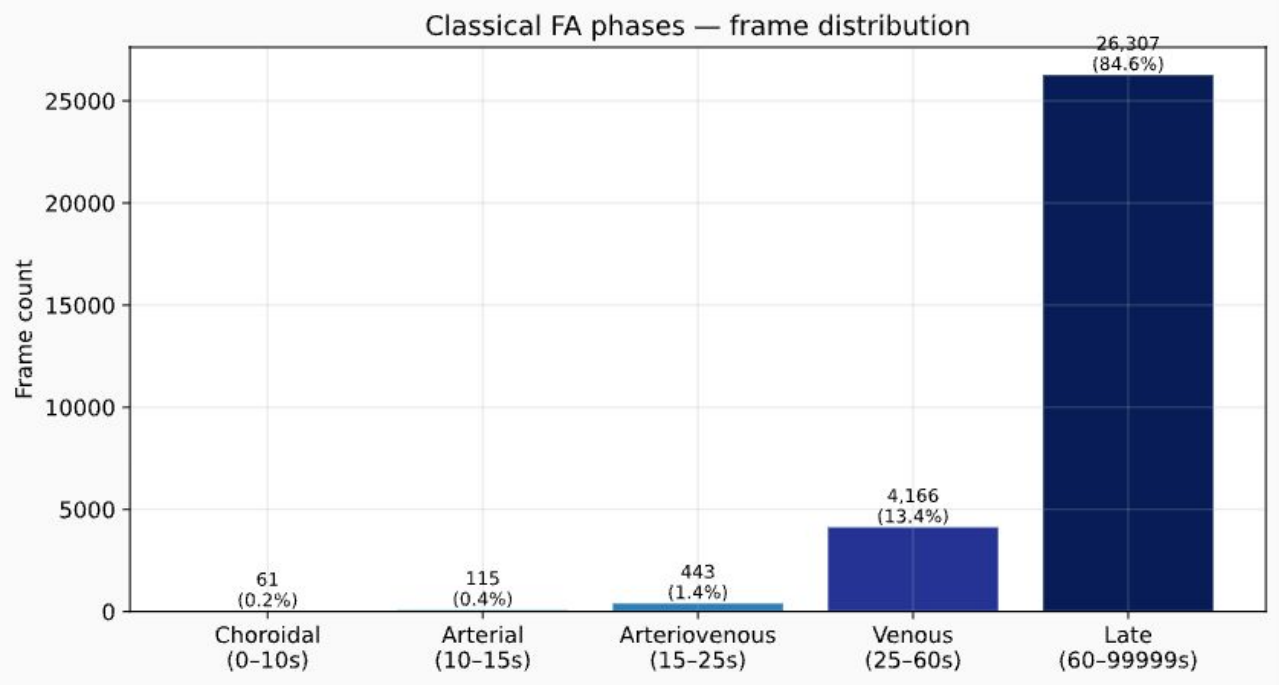
OCR; Results


When does each exam end?
Median = 14.0 min
Exam count
End time (min)
0
20
40
60
80
100
120
0
5
10
15
20
25
30
35

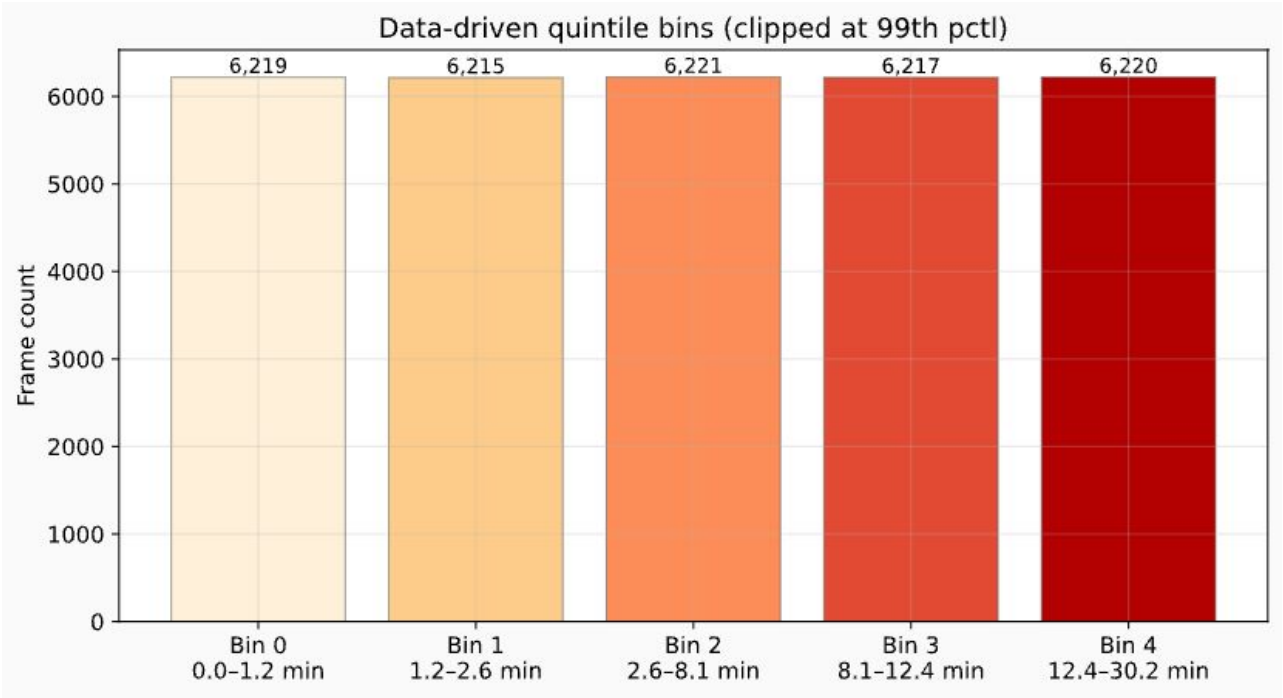
OCR; Results


Distribution of all frame timestamps (clipped at 99th pctl)
Quintile edges
1.2 min
Frame count
2500
2000
1500
1000
500
0
0
5
10
15
20
25
30
Elapsed time (min)

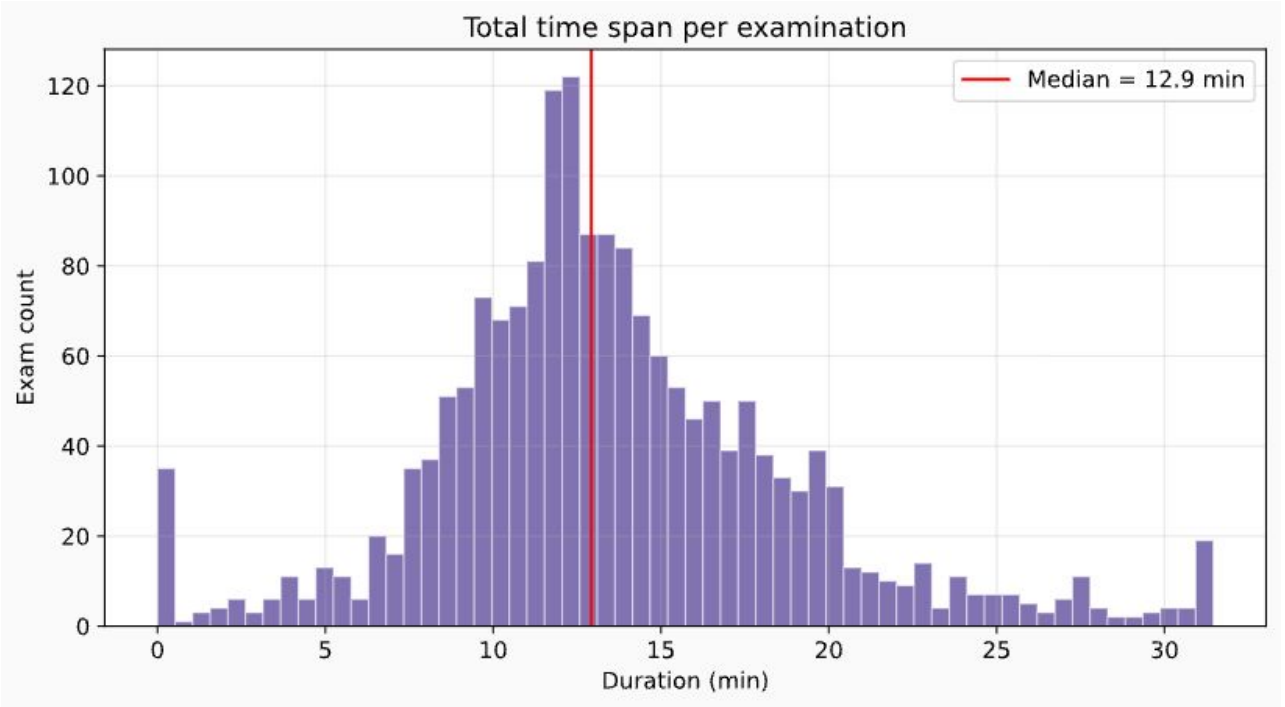
OCR; Results


Classical FA phases — frame distribution
Frame count
0
5000
10000
15000
20000
25000
61
(0.2%)
115
(0.4%)
443
(1.4%)
4,166
(13.4%)
26,307
(84.6%)
Choroidal
(0–10s)
Arterial
(10–15s)
Arteriovenous
(15–25s)
Venous
(25–60s)
Late
(60–99999s)

OCR; Results


Data-driven quintile bins (clipped at 99th pctl)
Frame count
0
1000
2000
3000
4000
5000
6000
6,219
6,215
6,221
6,217
6,220
Bin 0
0.0–1.2 min
Bin 1
1.2–2.6 min
Bin 2
2.6–8.1 min
Bin 3
8.1–12.4 min
Bin 4
12.4–30.2 min

OCR; Results


Total time span per examination
Median = 12.9 min
Exam count
Duration (min)
0
20
40
60
80
100
120
0
5
10
15
20
25
30

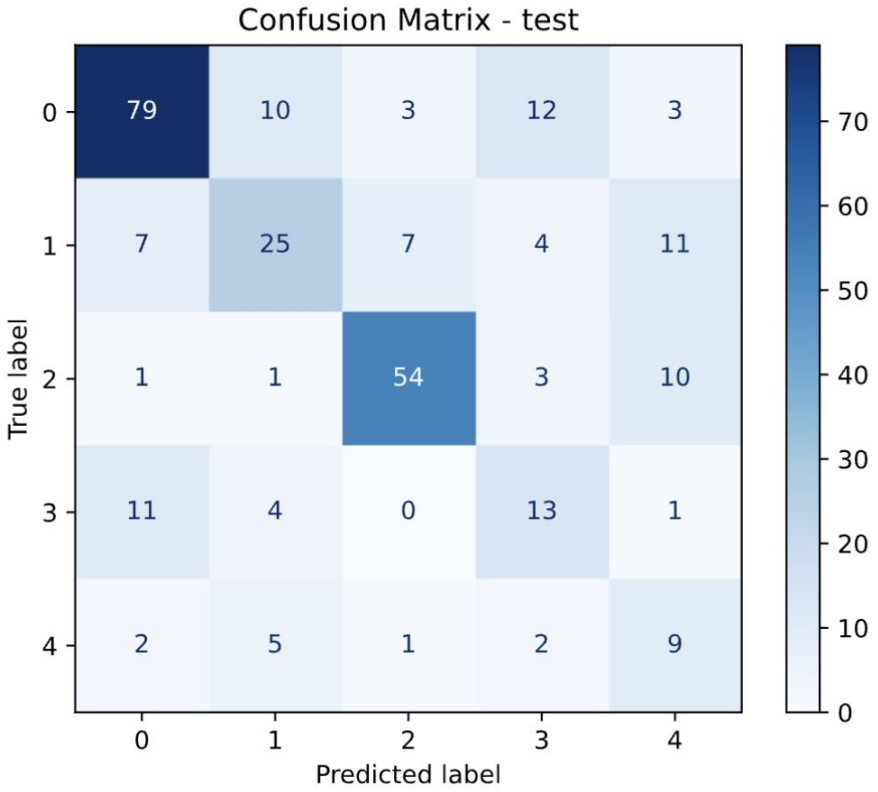
New baseline

Evaluation by using the best trained model

New baseline

Class	AUC	AP
Leakage	0.87	0.76
Staining	0.79	0.56
None	0.95	0.89
Pooling	0.77	0.33
Window Defect	0.79	0.23

New baseline



Last 2 frames (avg) vs. (new) Baseline

Last 2 frames (avg)

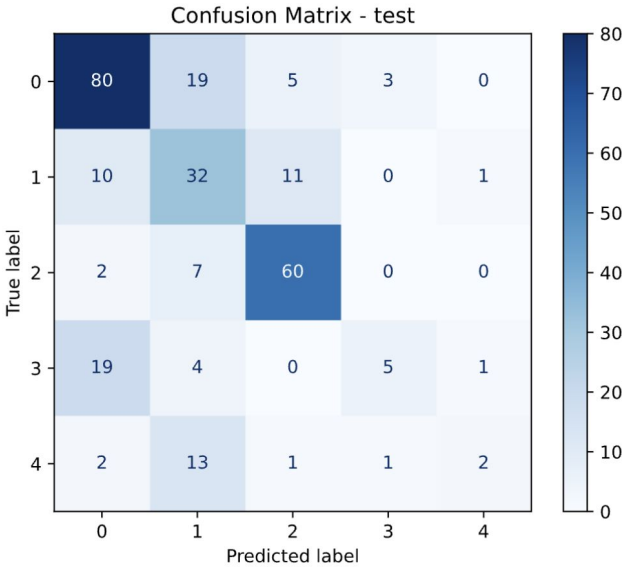
Class	AUC	AP
Leakage	0.85	0.73
Staining	0.75	0.40
None	0.95	0.83
Pooling	0.78	0.38
Window Defect	0.90	0.46

(new) Baseline

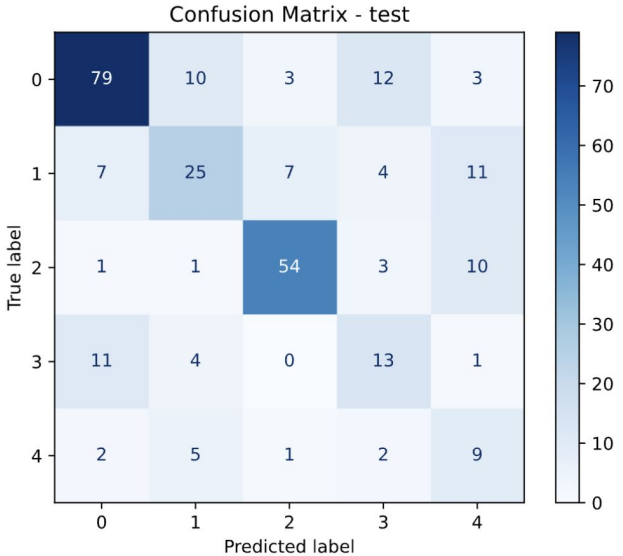
Class	AUC	AP
Leakage	0.87	0.76
Staining	0.79	0.56
None	0.95	0.89
Pooling	0.77	0.33
Window Defect	0.79	0.23

Last 2 frames (avg) vs. (new) Baseline

Last 2 frames (avg)



(new) Baseline



Frames to Videos _v1

34 frames

Pooling

As shown in the video, it is hard to manually find exams where the frames go gradually to the “important” frame.

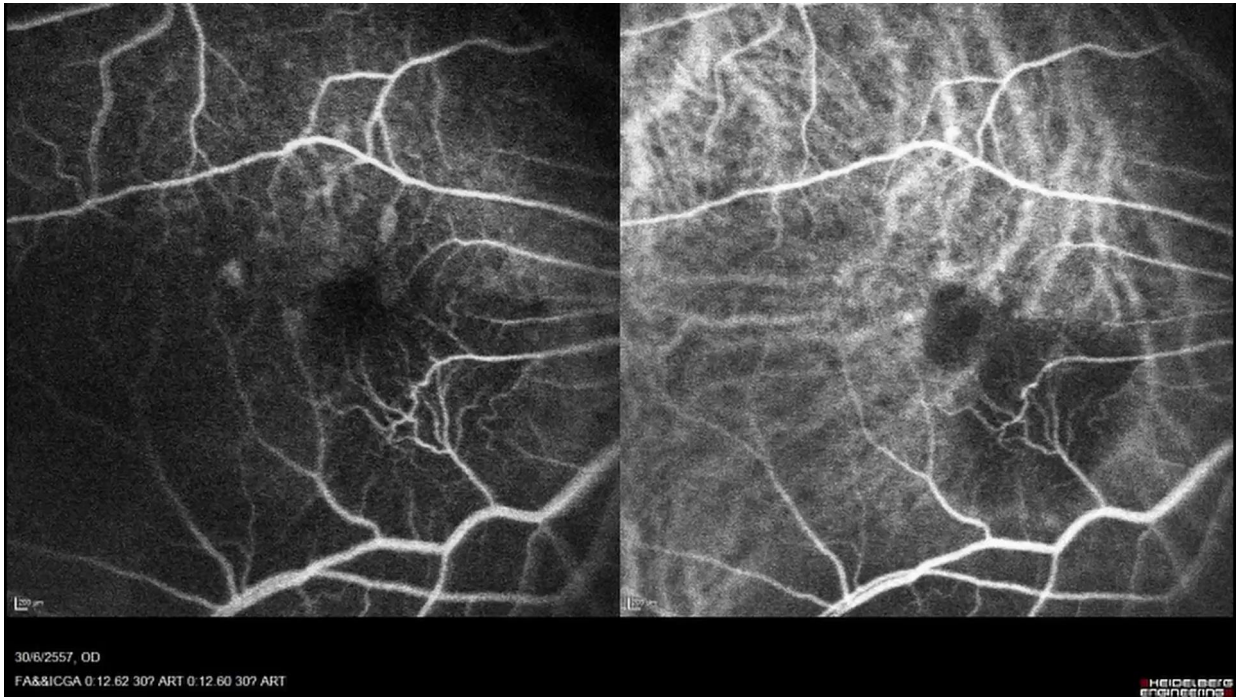
Which makes it hard to create useful video sequences.

Leakage

As shown in the video, it is hard to manually find exams where the frames go gradually to the “important” frame.

Which makes it hard to create useful video sequences.

38 Frames



None

As shown in the video, it is hard to manually find exams where the frames go gradually to the “important” frame.

Which makes it hard to create useful video sequences.

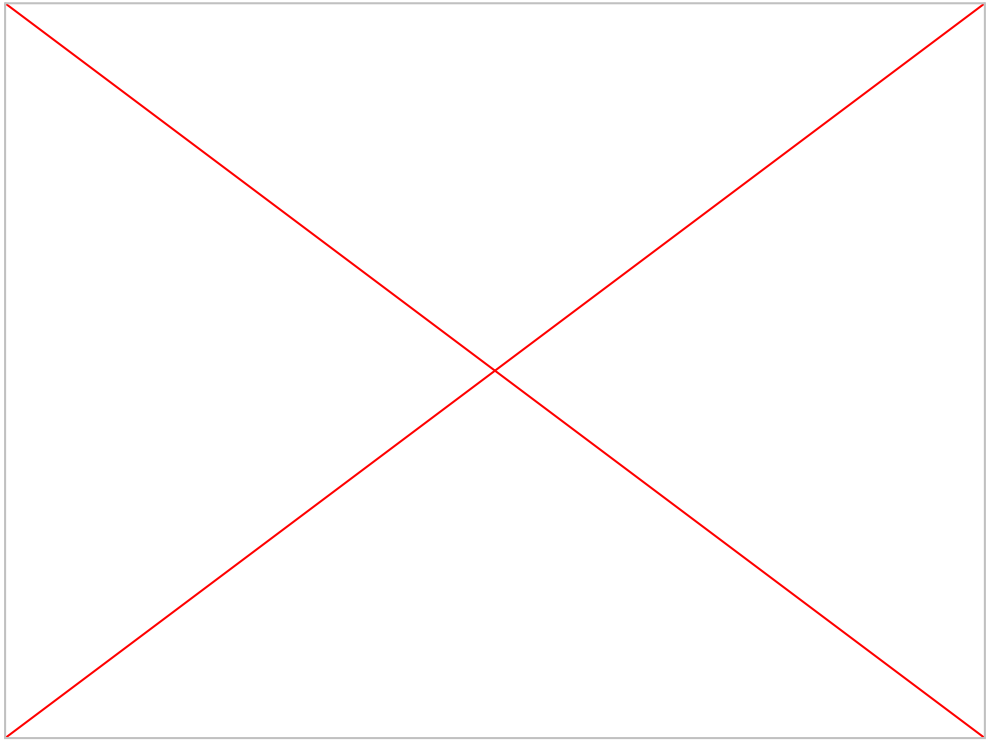
32 Frames

Staining

As shown in the video, it is hard to manually find exams where the frames go gradually to the “important” frame.

Which makes it hard to create useful video sequences.

39 Frames

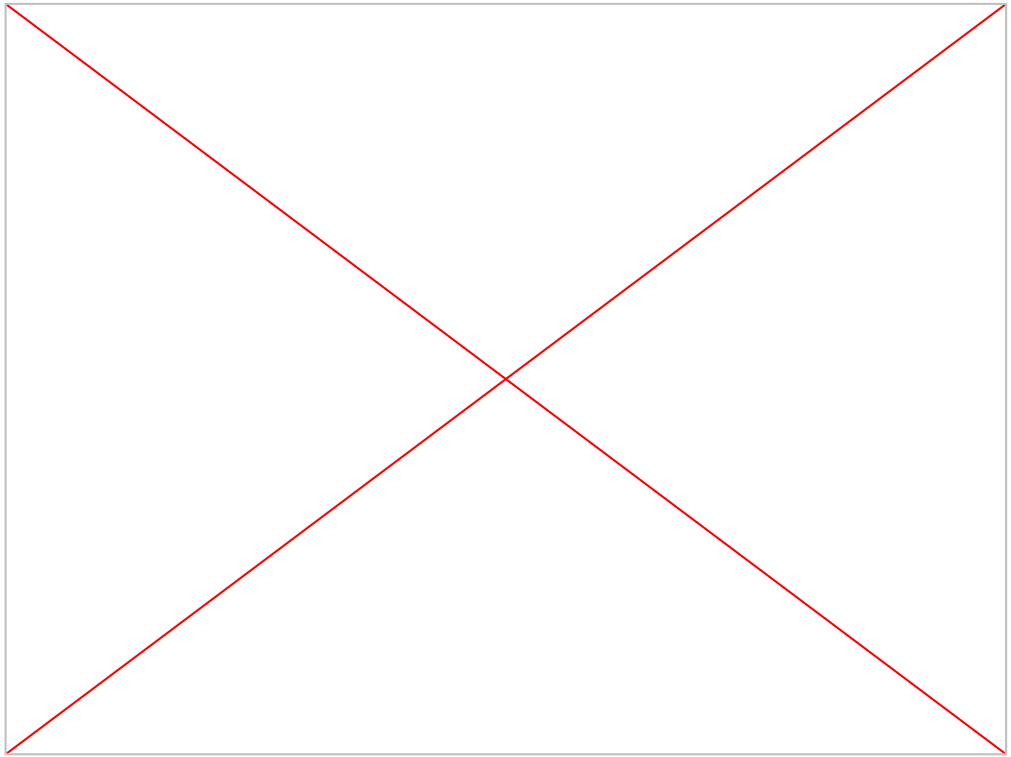


Window Defect

As shown in the video, it is hard to manually find exams where the frames go gradually to the “important” frame.

Which makes it hard to create useful video sequences.

36 Frames



Conclusion

I do believe having actual good video sequences of the FA exams could be useful, but it would probably be better to manually pick out the good frames and check across a lot of exams, which I did not think was worth the time for now.

If this is truly important I could give it another shot and take some more time trying to get a better representative video for each HyperF_Type.

Not quite happy with the results as of right now.

Study

Attention-based Deep Multiple Instance Learning
Link

Study; Key takeaways

They compare mean pooling (which I did in week 1) with attention pooling.

- The attention pooling seemed to be better than mean or max pooling, because not all instances are equally informative
- Attention weights are interpretable, they can visualize which frames get high weights

Study; Our case

It will show:

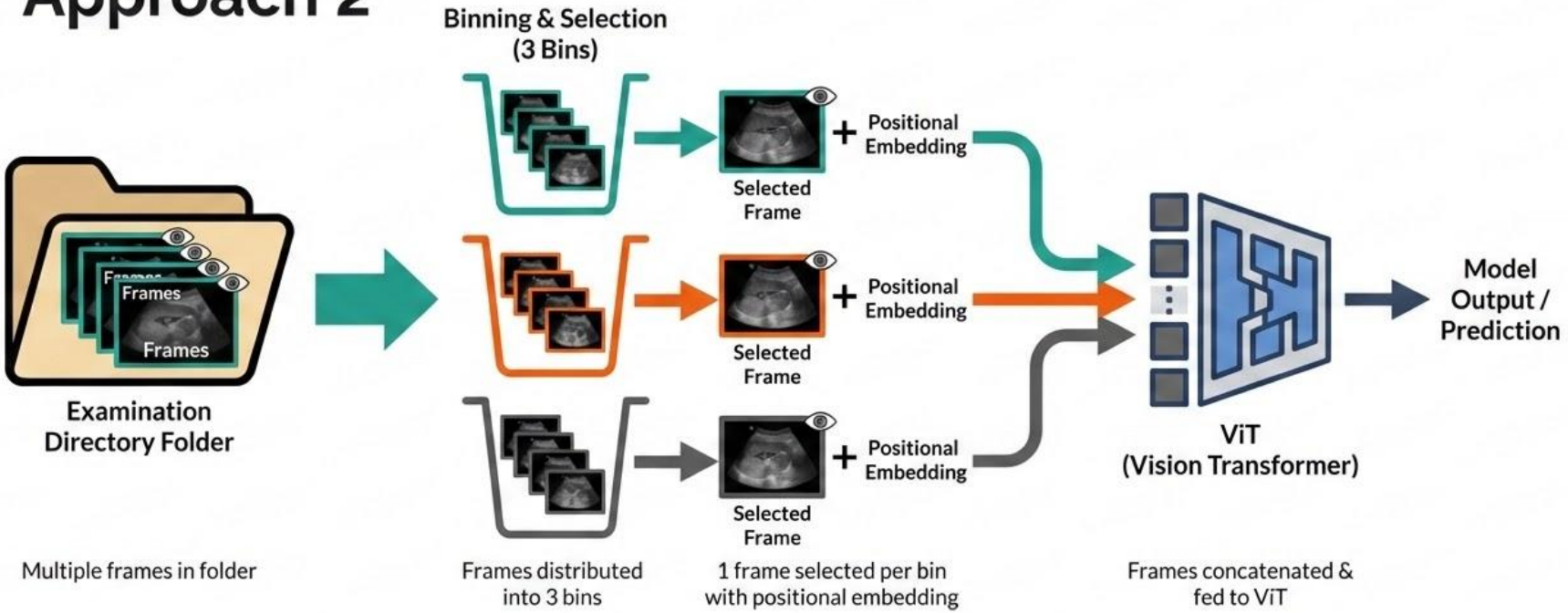
- For x, attention peaks late.
- For y, attention peaks early.
- Fits well into the binning approach, to see which bin is most important, and it’s not just “dumb” averaging anymore.

What probably will happen:

- The model will probably learn to put bigger weights on the last bin(s), because the most information will probably be present here. But it’s worth a try to see it for ourselves! (Not a bad thing by the way, still takes in multiple frames).

Using bins to get “medical intuition”

Approach 2

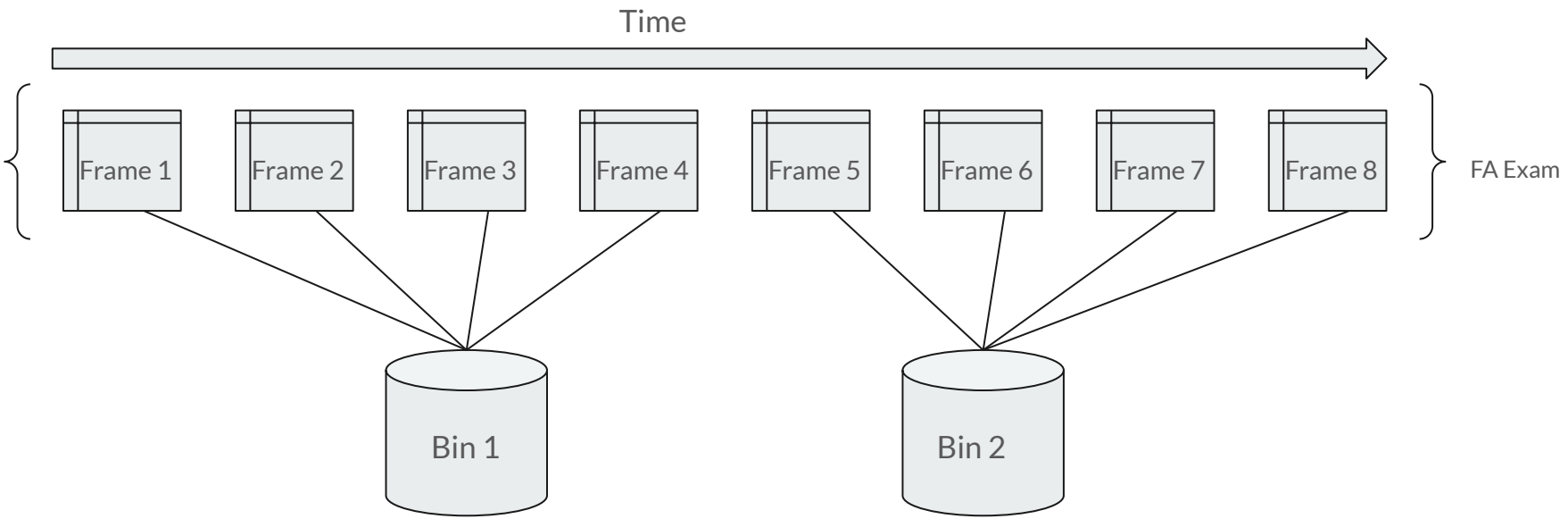


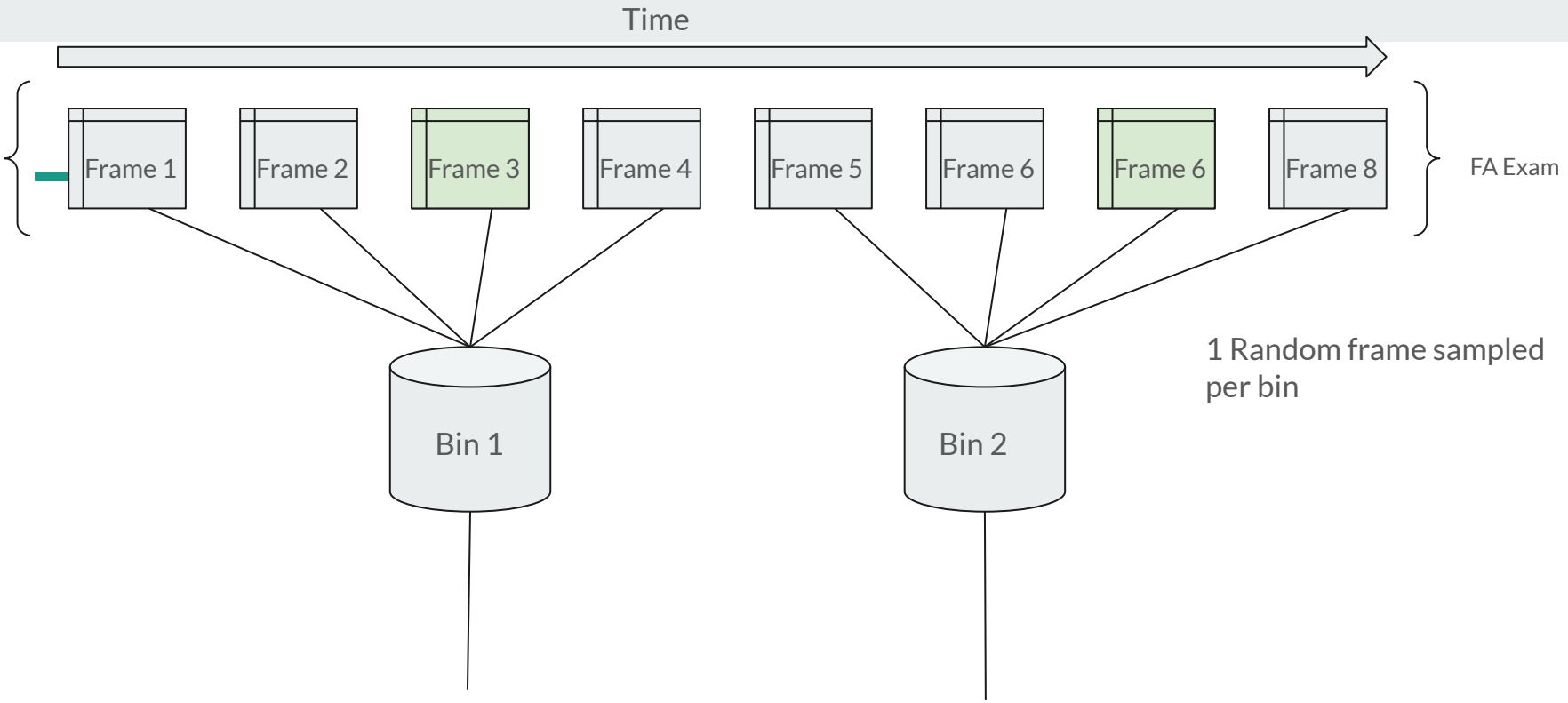
Approach 2; Motivation – Why bins?

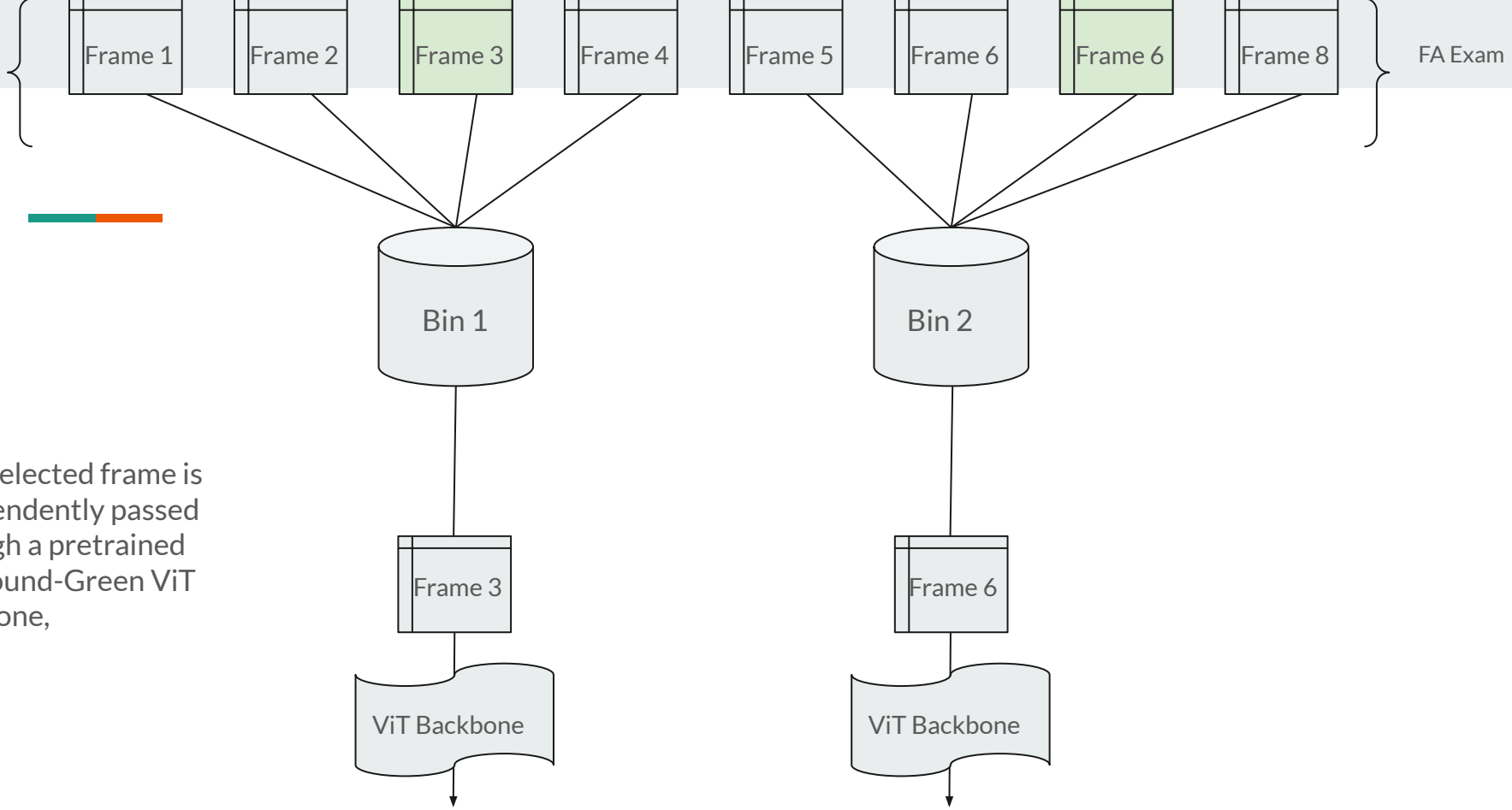
Current approaches use only the last frame or average last_n frames, losing temporal information about dye progression

Single Frame	**Averaged Frames**	**Bins Approach**
1 image (latest frame)	Last_n frames -> blurry average	K frames out of K time periods
“Ignores big part of exam”	“Destroys temporal information”	“Captures Progression”

Approach 2; Binning Strategy (n_bins = 2)



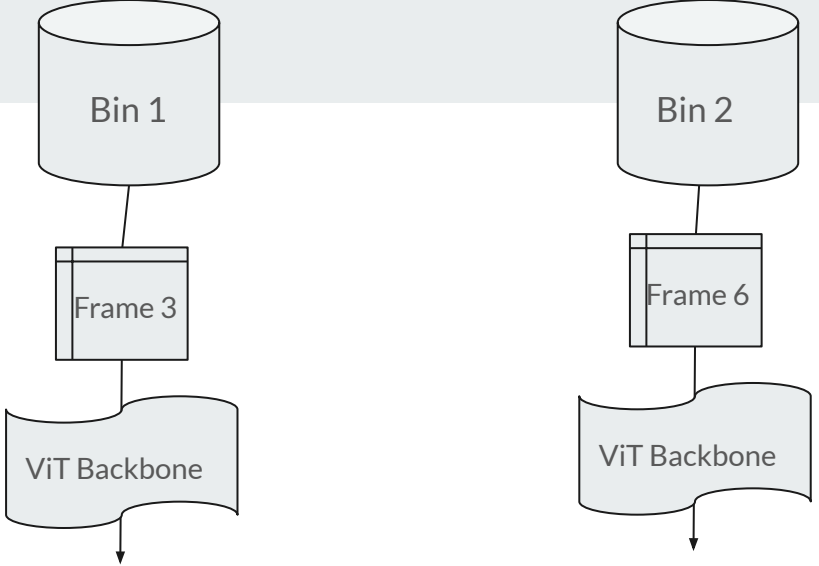

Time
Frame 1
Frame 2
Frame 3
Frame 4
Frame 5
Frame 6
Frame 6
Frame 8
FA Exam
Bin 1
Bin 2
1 Random frame sampled per bin



Each selected frame is independently passed through a pretrained RETFound-Green ViT backbone,

producing a 384-dimensional feature vector per frame.

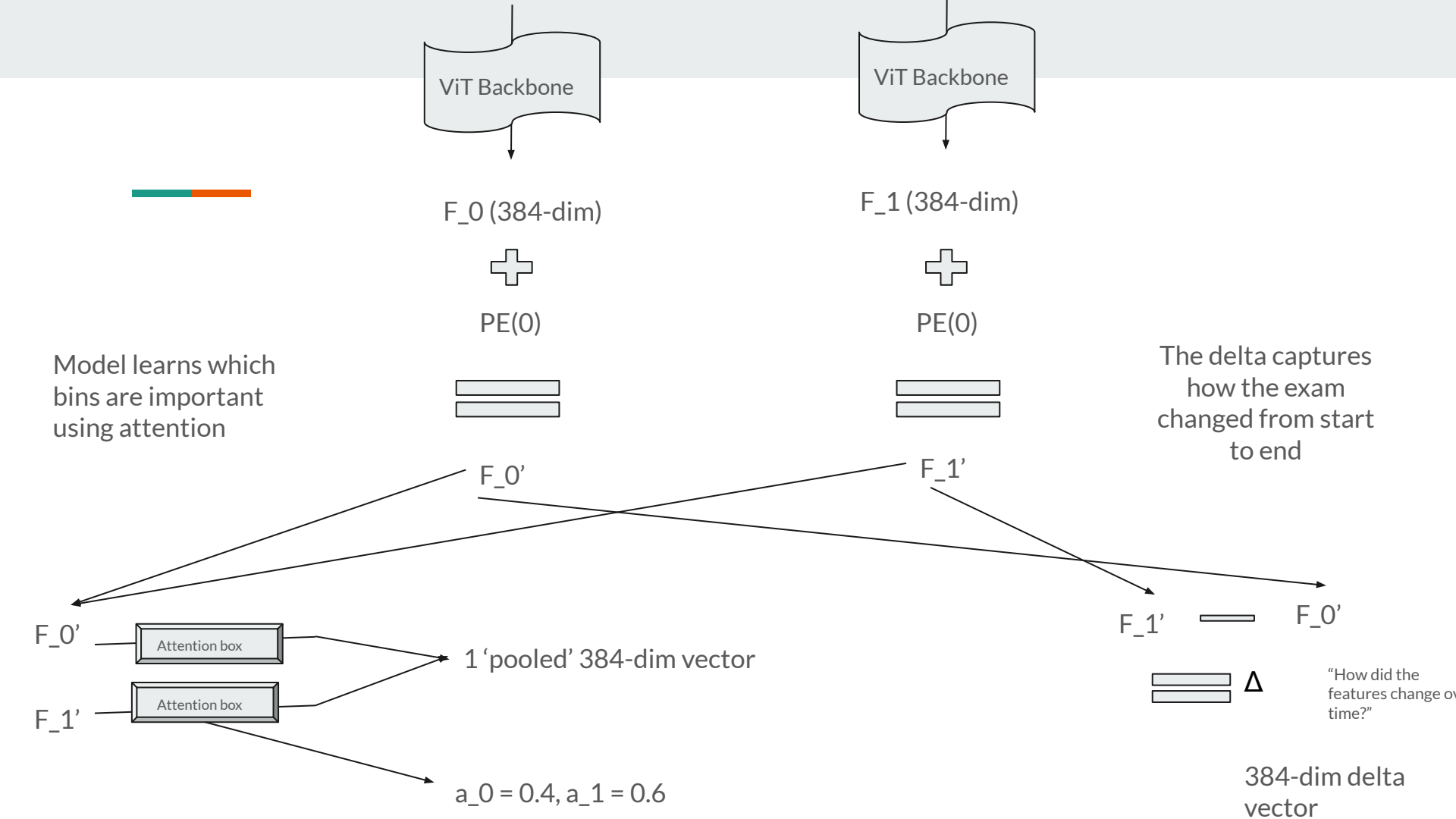
Learnable positional embeddings are added to each feature vector, encoding the chronological position of each bin.

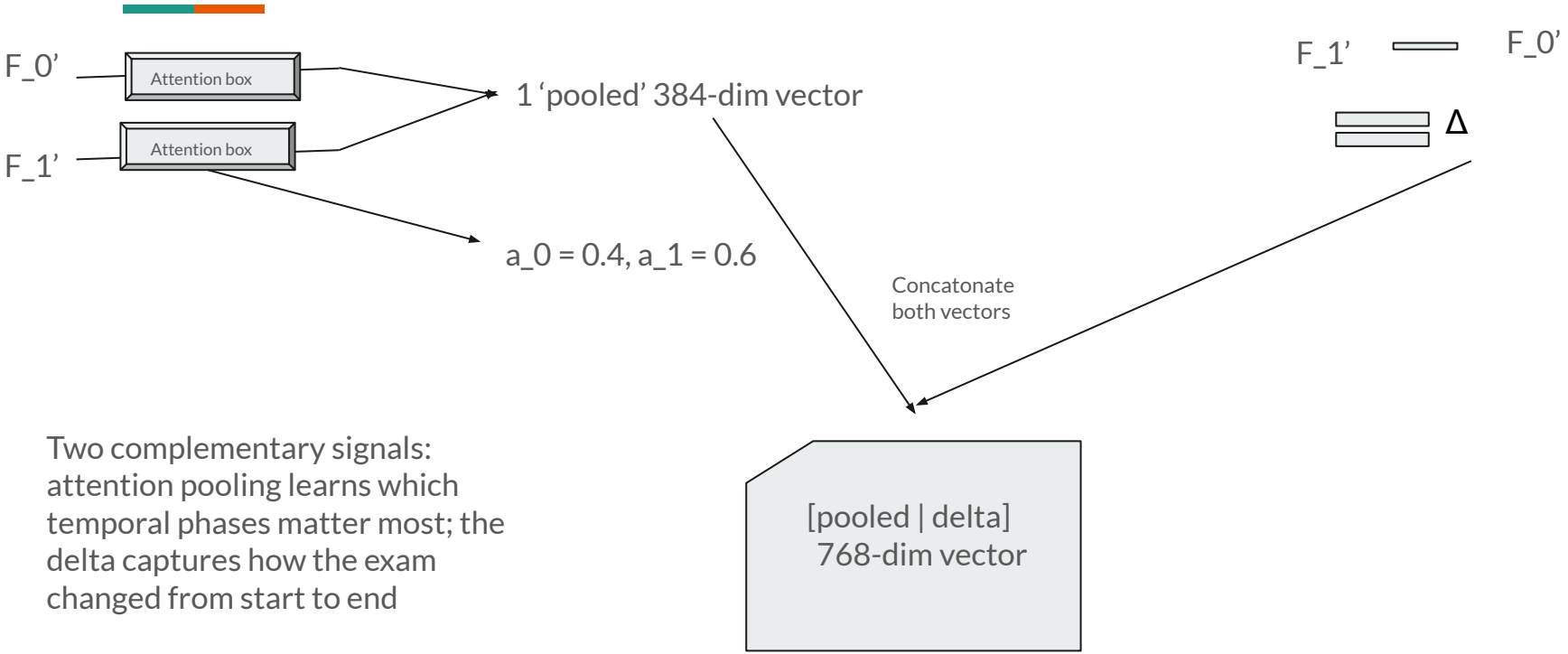


Knows it's early

Knows it's late

This allows the model to distinguish early from late frames


ViT Backbone
ViT Backbone
F_0 (384-dim)
F_1 (384-dim)
PE(0)
PE(0)
Model learns which bins are important using attention
The delta captures how the exam changed from start to end
F_0'
F_1'
F_0'
Attention box
Attention box
F_1'
1 'pooled' 384-dim vector
a_0 = 0.4, a_1 = 0.6
F_1'
F_0'
Δ
"How did the features change ov
time?"
384-dim delta vector



Two complementary signals: attention pooling learns which temporal phases matter most; the delta captures how the exam changed from start to end

Bins(n=3) vs. (new) Baseline

Bins(n=3)

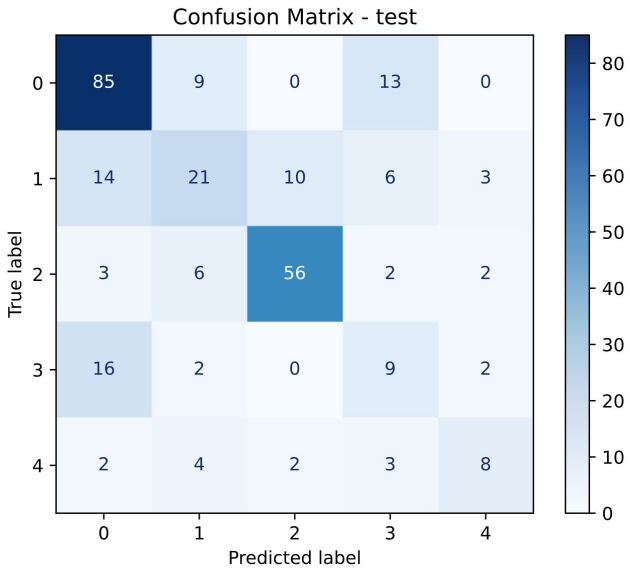
Class	AUC	AP
Leakage	0.87	0.73
Staining	0.73	0.41
None	0.94	0.83
Pooling	0.71	0.34
Window Defect	0.84	0.47

(new) Baseline

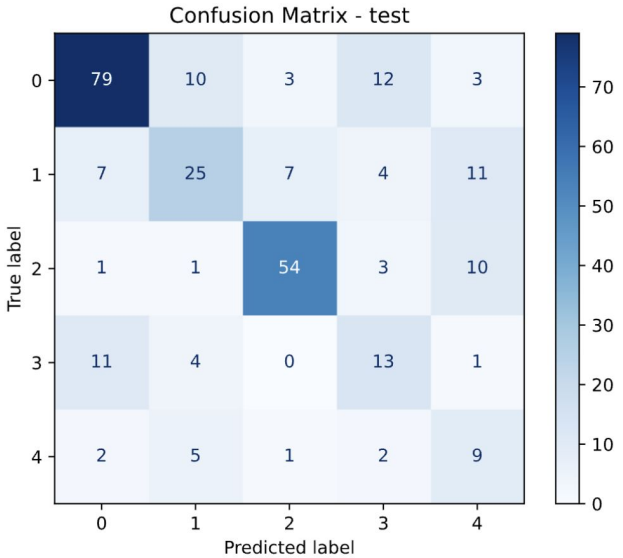
Class	AUC	AP
Leakage	0.87	0.76
Staining	0.79	0.56
None	0.95	0.89
Pooling	0.77	0.33
Window Defect	0.79	0.23

Bins(b=3) vs. (new) Baseline

Bins(n=3)



(new) Baseline



Bins(n=3) vs. Last 2 frames (avg)

Bins(n=3)

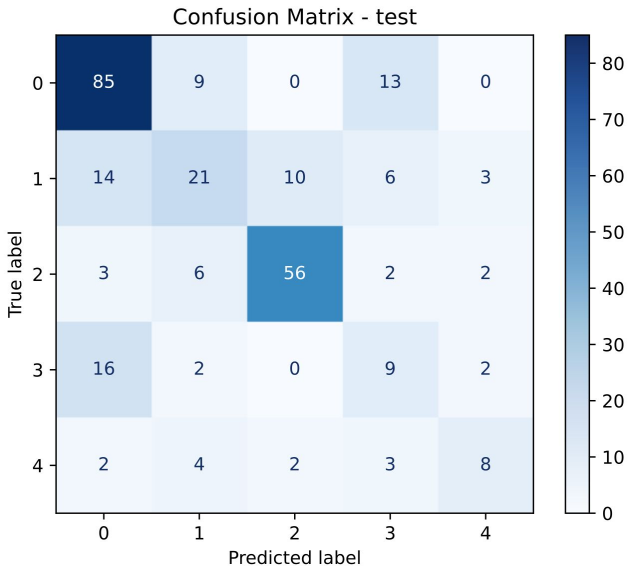
Class	AUC	AP
Leakage	0.87	0.73
Staining	0.73	0.41
None	0.94	0.83
Pooling	0.71	0.34
Window Defect	0.84	0.47

Last 2 grames (avg)

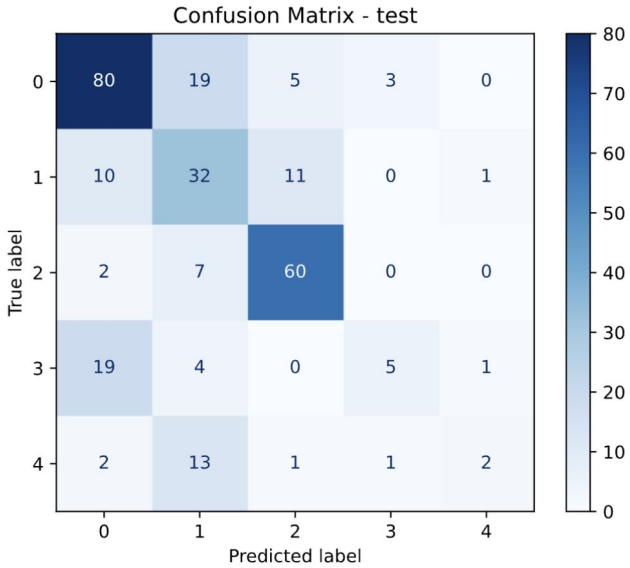
Class	AUC	AP
Leakage	0.85	0.73
Staining	0.75	0.40
None	0.95	0.83
Pooling	0.78	0.38
Window Defect	0.90	0.46

Bins(b=3) vs. Last 2 frames (avg)

Bins(n=3)



Last 2 frames (avg)



Bins(n=3) vs. Bins(n=3 + delta)

Bins(n=3)

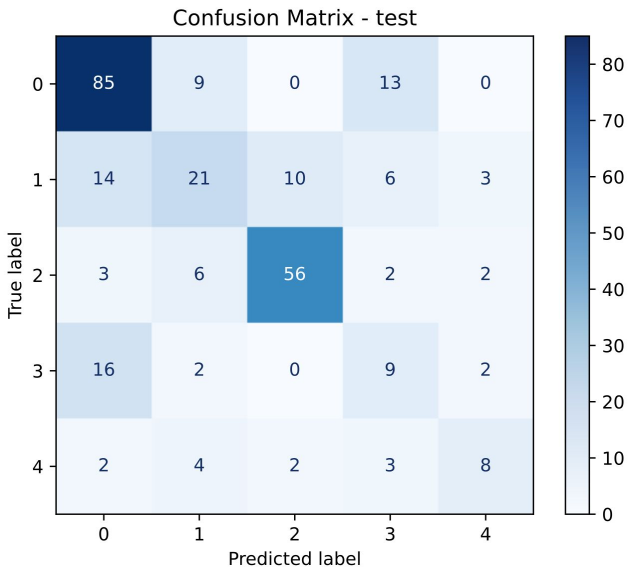
Class	AUC	AP
Leakage	0.87	0.73
Staining	0.73	0.41
None	0.94	0.83
Pooling	0.71	0.34
Window Defect	0.84	0.47

Bins(n=3 + delta)

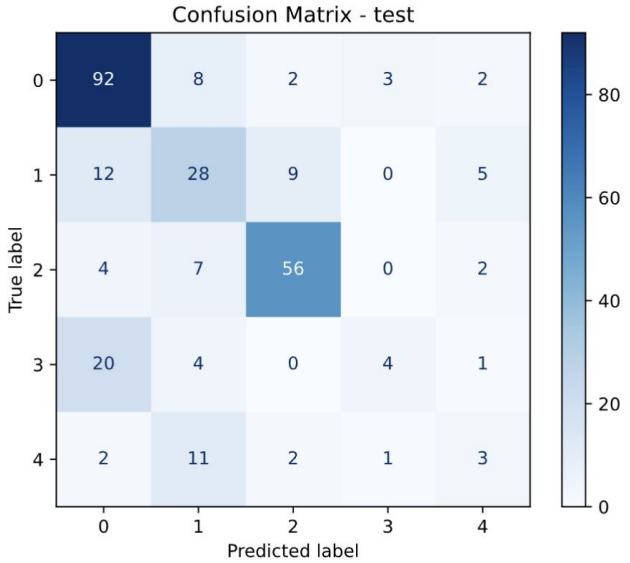
Class	AUC	AP
Leakage	0.88	0.76
Staining	0.82	0.51
None	0.94	0.84
Pooling	0.75	0.34
Window Defect	0.84	0.24

Bins(n=3) vs. Bins(n=3 + delta)

Bins(n=3)



Bins(n=3 + delta)



Bins(n=5 + delta) vs. Bins(n=3 + delta)

Bins(n=5 + delta)

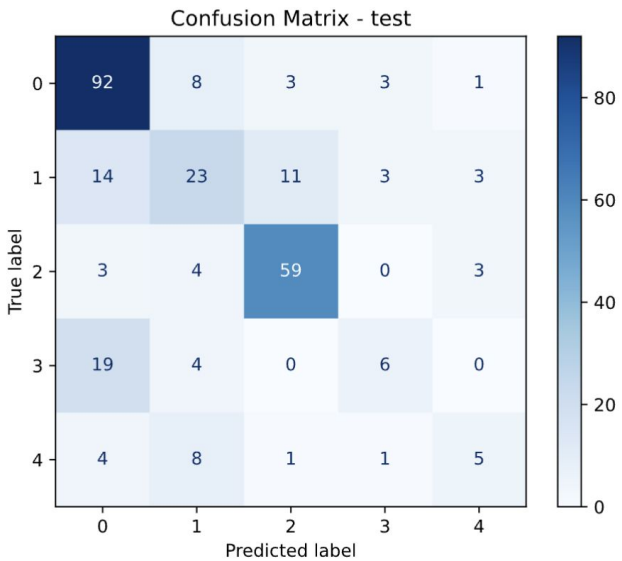
Class	AUC	AP
Leakage	0.89	0.81
Staining	0.80	0.48
None	0.97	0.90
Pooling	0.77	0.35
Window Defect	0.88	0.38

Bins(n=3 + delta)

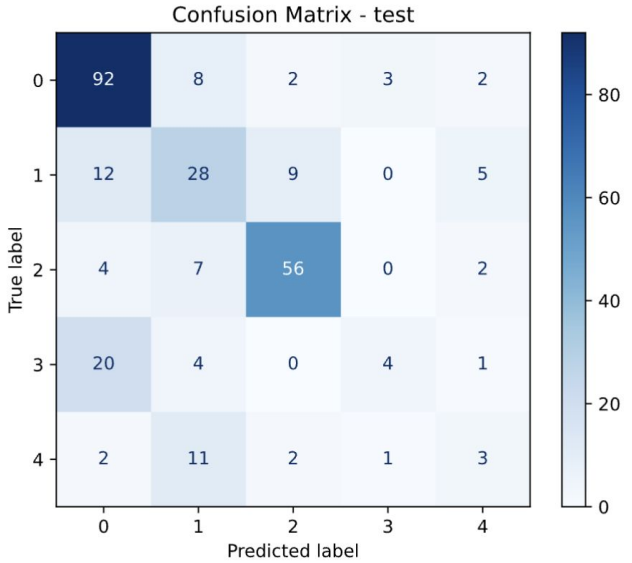
Class	AUC	AP
Leakage	0.88	0.76
Staining	0.82	0.51
None	0.94	0.84
Pooling	0.75	0.34
Window Defect	0.84	0.24

Bins(n=3) vs. Bins(n=3 + delta)

Bins(n=5 + delta)



Bins(n=3 + delta)



Bins(n=5 + delta) vs. (new) Baseline

Bins(n=5 + delta)

Class	AUC	AP
Leakage	0.89	0.81
Staining	0.80	0.48
None	0.97	0.90
Pooling	0.77	0.35
Window Defect	0.88	0.38

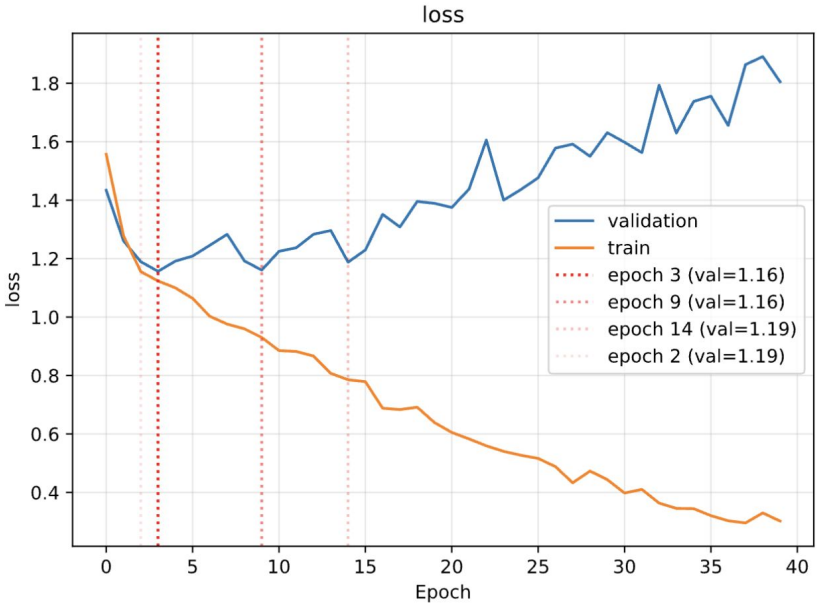
(new) Baseline

Class	AUC	AP
Leakage	0.87	0.76
Staining	0.79	0.56
None	0.95	0.89
Pooling	0.77	0.33
Window Defect	0.79	0.23

Winner: Bins(n=5 + delta) but...

Overfitting occurs when fine tuning the entire backbone.

So next step is to partially freeze the RETFound-Green backbone. Only the last 3 blocks of the transformer can now learn features for our task, might give improvements.



Bins(n=5 + delta) vs. Bins(n=5 + delta + 6 partial freeze)

Bins(n=5 + delta)

Class	AUC	AP
Leakage	0.89	0.81
Staining	0.80	0.48
None	0.97	0.90
Pooling	0.77	0.35
Window Defect	0.88	0.38

Bins(n=5 + delta + 6 partial freeze)

Class	AUC	AP
Leakage	0.91	0.82
Staining	0.80	0.48
None	0.96	0.89
Pooling	0.85	0.45
Window Defect	0.87	0.39

Next week(s)

Next Week(s)

Ideas:

- The bin picking is still random, could perform better with some actual logic
- Nothing is done with time stamps as of now, possibly try to implement that into the binning logic
- Implement Laplacian frame selection for bins
- Replace BinAttentionPool for 1-2 layers of nn.TransformerEncoderLayer follow by a CLS pooling
- More frames being used per bin