Project 2

Ball action spotting

FINAL PRESENTATION

Team 3

C6 - Video Analysis @ UB

Andrea Sánchez
Daniel Pardo
Merlès Subirà
Francisco de Paula Urmeneta

24th April 2025

Index

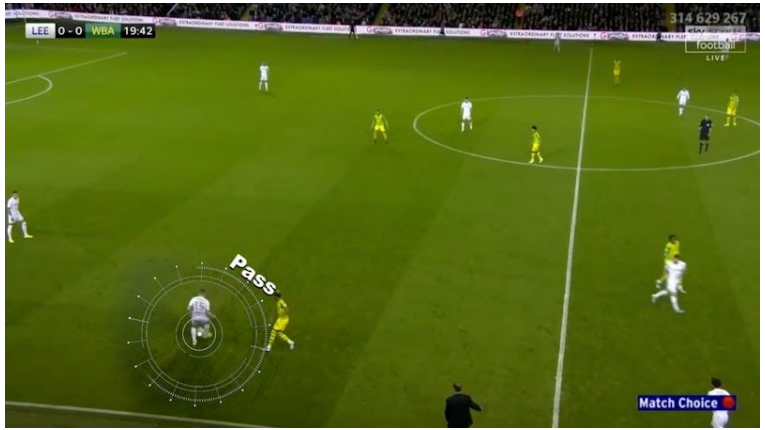
1.a. Ball action spotting: Dataset

SoccerNet
Dataset and Challenges

- 7 full football matches (~15GB)
  - 4 for training
  - 1 for validation
  - 2 for testing
- **Unbalanced** in the number of appearances per class

12 ACTIONS + 1

pass (4703)
drive (4067)
header (685)
high pass (778)
out (590)
cross (320)
throw in (392)
shot (190)
ball player block (220)
player successful tackle (73)
free kick (26)
goal (14)
no-action(487942)

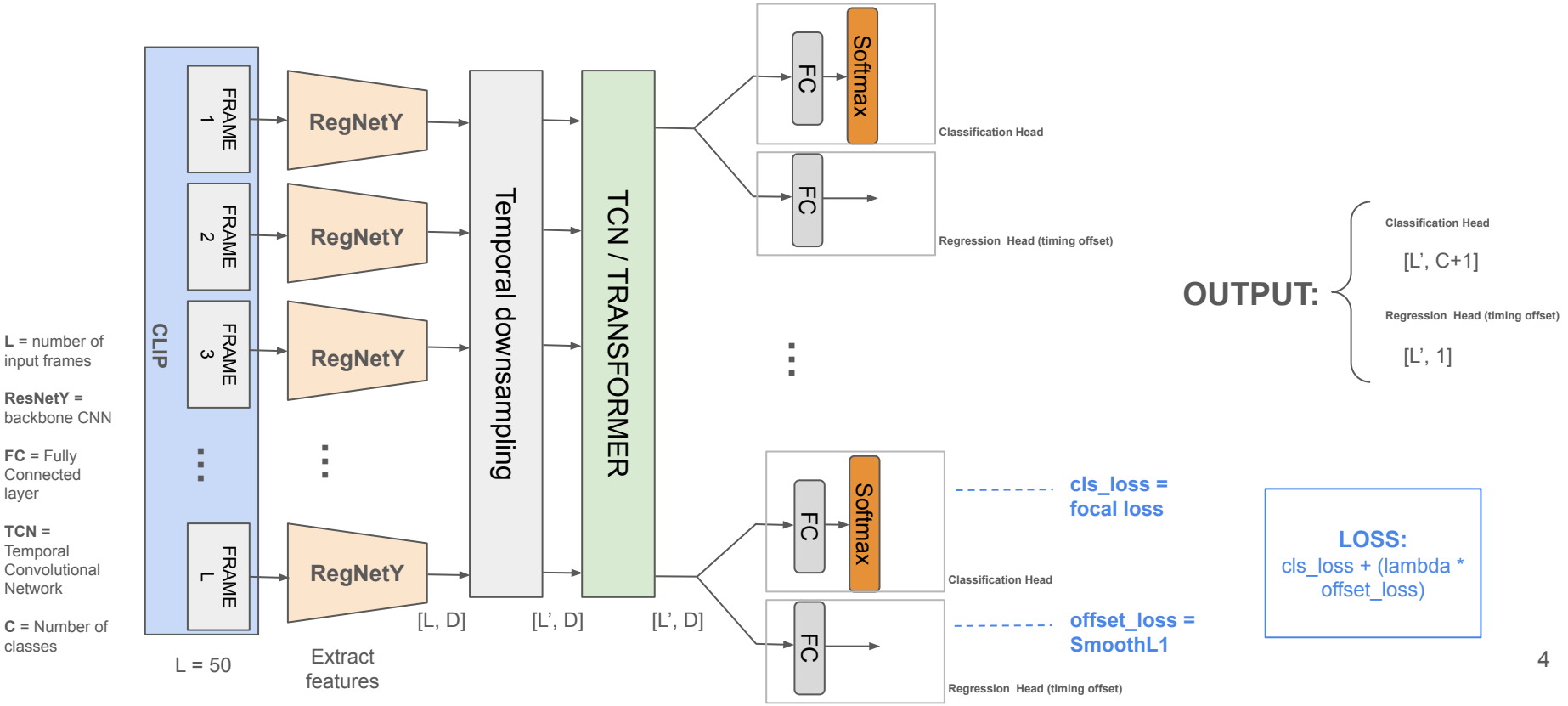


frame annotation format:

```
{
    "gameTime": "1 - 00:01",
    "label": "PASS",
    "position": "1120",
    "team": "left",
    "visibility": "visible"
},
```

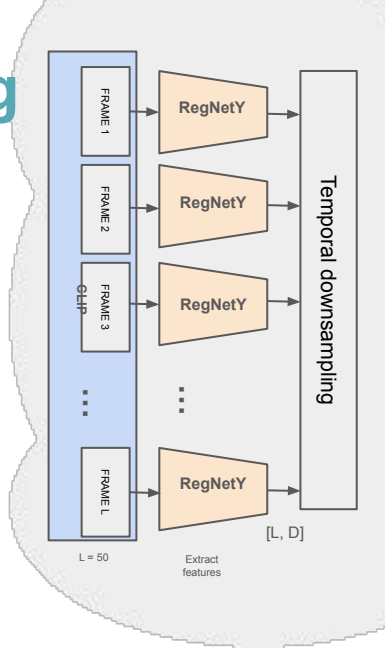
1.b. Ball action spotting: Proposed pipeline

Goal: predict the exact timestamps when key ball-related actions occur in soccer videos. We assume that each frame can correspond to only one action, or no action at all.



1.b. Ball action spotting: Temporal downsampling

Purpose: Reduce the temporal resolution of feature sequences in order to capture higher-level temporal patterns and improve computational efficiency.



INPUT → **[L, D]**

with L' < L

OUTPUT → **[L', D]**

✅

- **Lower temporal resolution** → higher-level temporal patterns.
- Reduces **redundancy**
- Improves computational **efficiency.**
- Can reduce overfitting to frame-level noise.

⚠️

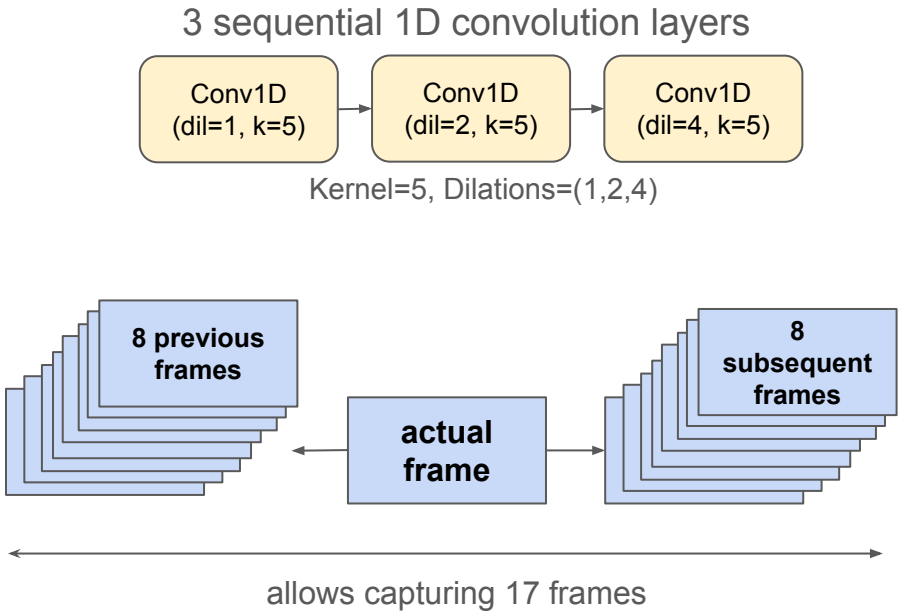
- **Loss of precision** → localizing on fine-grained events.
- With a **stride of 1**, too many redundant frames, with a **larger stride,** important context may be lost.
- Can introduce **noise or overfitting**, especially due to class imbalance (events are sparse).

Temporal downsampling → we need a **displacement head** that predicts the temporal offset to the actual event. Allows the model to recover frame-level precision despite the reduced temporal resolution.

1.b. Ball action spotting: TCN / Transformer

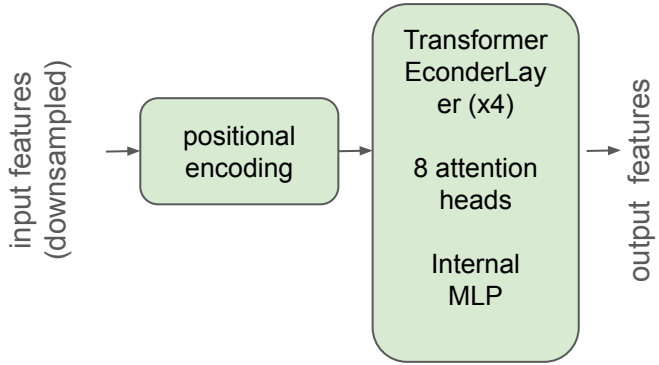
Temporal Convolutional Network (TCN)

Allows us to model the temporality of frames with a **centered time window**. It doesn't view the entire sequence as a transformer, but it is **simpler** and **easier to train**.

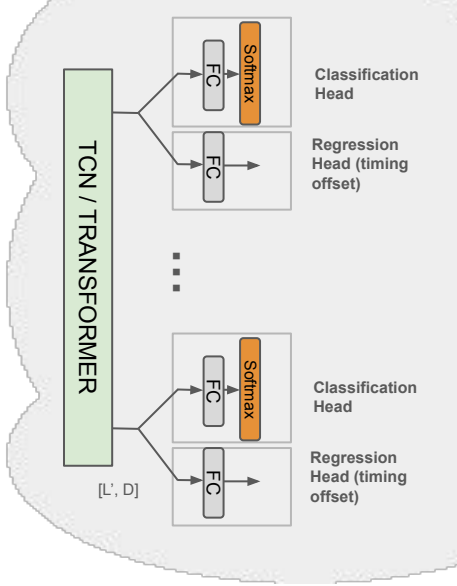


Transformer

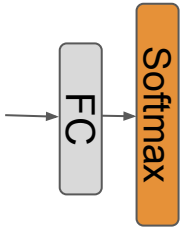
We add a transformer with **multi-head self-attention**. Allows us to capture temporal dependencies between frames. Each frame takes care of **itself** and the **rest of the frames** in the sequence.



1.b. Ball action spotting: Final Heads

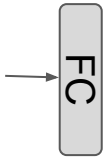


Classification Head



gives the class probability

Regression Head for offset (displacement head)



- Temporal downsampling **allows training a regression head** to correct temporal misalignment (unlike initial stride at clip level).
- This offset head **learns** how many frames ahead or behind the anchor frame the event actually occurs.

How it works:

- During **training** we **create ground truth (GT) offset labels** per batch.
- **Temporal downsampling** reduces the number of frames the model sees.
- We **calculate the offset** between these visible frames and the actual event frame from the original sequence.

1.b. Ball action spotting: Temporal Offset GT

Example: given original labels of:

- Clip with 50 frames.
- Labels with events.
- Temporal downsampling with stride 2.

The model sees the red labels (25 frames).

```
2, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 4, 0, 0, 0, 0, 0, 0
```

Labels used for training (blue):

```
2, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
0, 0, 0, 0, 0
```

Create offset GT for downsampled labels by using original labels to compute the **distance to the nearest event**.

```
0.,  -2.,  -4.,  -6.,  -8., -10., -12., -14.,  14.,
12.,  10.,   8.,  6.,   4.,   2.,   0.,  -2.,  -4.,
-6.,   5.,   3.,   1.,  -1.,  -3., -5.
```

Improvement: apply a **mask** to ignore frames too far from events when training the offset head, but keep all for classification. We set the window to 8 frames.

```
0., -2., -4., -6., -8., nan, nan, nan, nan, nan, nan,
8.,  6.,  4., 2.,  0., -2., -4., -6.,  5.,  3.,  1.,
-1., -3., -5.],
```

✅ Offset is only predicted/trained near events → better learning, model focuses on important moments, less noise.

⚠️ Adds a new hyperparameter (window size, set to 8), which ideally should be tuned too, but we didn't have time to.

1.b. Ball action spotting: Temporal Offset Correction

Training: This offset is not used to shift predictions, it's only used as part of the **loss function** to **train** the model. The model learns to predict both the **class** of the action and the **temporal offset** to the true frame.

Inference: Here, the offset is actually used to correct the prediction. Since downsampling reduces the number of frames, the **corrected prediction frame** is calculated as:

corr_pos = round(pred_index * stride + offset)

Example: video segment with 6 input frames and two classes.

Apply temporal downsampling with a **stride of 2**, resulting in 3 predictions.

Each prediction includes both a classification score and a temporal offset estimating how far the predicted frame is from the true event frame.

Frame	**Classification**		**Offset**
	Class 0	**Class 1**	
0	0.1	0.9	0.0
1	0.6	0.4	0.5
2	0.2	0.8	-0.5

1.b. Ball action spotting: Temporal Offset Correction

Training: This offset is not used to shift predictions, it's only used as part of the **loss function** to **train** the model. The model learns to predict both the **class** of the action and the **temporal offset** to the true frame.

Inference: Here, the offset is actually used to correct the prediction. Since downsampling reduces the number of frames, the **corrected prediction frame** is calculated as:

corr_pos = round(pred_index * stride + offset)

Example: video segment with 6 input frames and two classes. Apply temporal downsampling with a **stride of 2**, resulting in 3 predictions. Each prediction includes both a classification score and a temporal offset estimating how far the predicted frame is from the true event frame.

Frame	Classification	
	Class 0	Class 1
0	0.1	0.9
1	0	0
2	0	0
3	0	0
4	0	0
5	0	0

Prediction 0

- pred_index = 0
- Position = 0×2 = 0
- Offset = 0.0
- Corrected = round(0+0.0)=0

→ Place [0.1, 0.9] at frame 0.

1.b. Ball action spotting: Temporal Offset Correction

Training: This offset is not used to shift predictions, it's only used as part of the **loss function** to **train** the model. The model learns to predict both the **class** of the action and the **temporal offset** to the true frame.

Inference: Here, the offset is actually used to correct the prediction. Since downsampling reduces the number of frames, the **corrected prediction frame** is calculated as:

corr_pos = round(pred_index * stride + offset)

Example: video segment with 6 input frames and two classes. Apply temporal downsampling with a **stride of 2**, resulting in 3 predictions. Each prediction includes both a classification score and a temporal offset estimating how far the predicted frame is from the true event frame.

Frame	Classification	
	Class 0	Class 1
0	0.1	0.9
1	0	0
2	0	0
3	0.6	0.4
4	0	0
5	0	0

Prediction 1

- i=1
- Position = 1×2=2
- Offset = 0.5
- Corrected = round(2+0.5)=3

→ Place [0.6, 0.4] at frame 3.

1.b. Ball action spotting: Temporal Offset Correction

Training: This offset is not used to shift predictions, it's only used as part of the **loss function** to **train** the model. The model learns to predict both the **class** of the action and the **temporal offset** to the true frame.

Inference: Here, the offset is actually used to correct the prediction. Since downsampling reduces the number of frames, the **corrected prediction frame** is calculated as:

corr_pos = round(pred_index * stride + offset)

Example: video segment with 6 input frames and two classes. Apply temporal downsampling with a **stride of 2**, resulting in 3 predictions. Each prediction includes both a classification score and a temporal offset estimating how far the predicted frame is from the true event frame.

Note: If multiple predictions are corrected to the **same frame**, their scores are added. This ensures that no information is lost due to overlapping predictions.

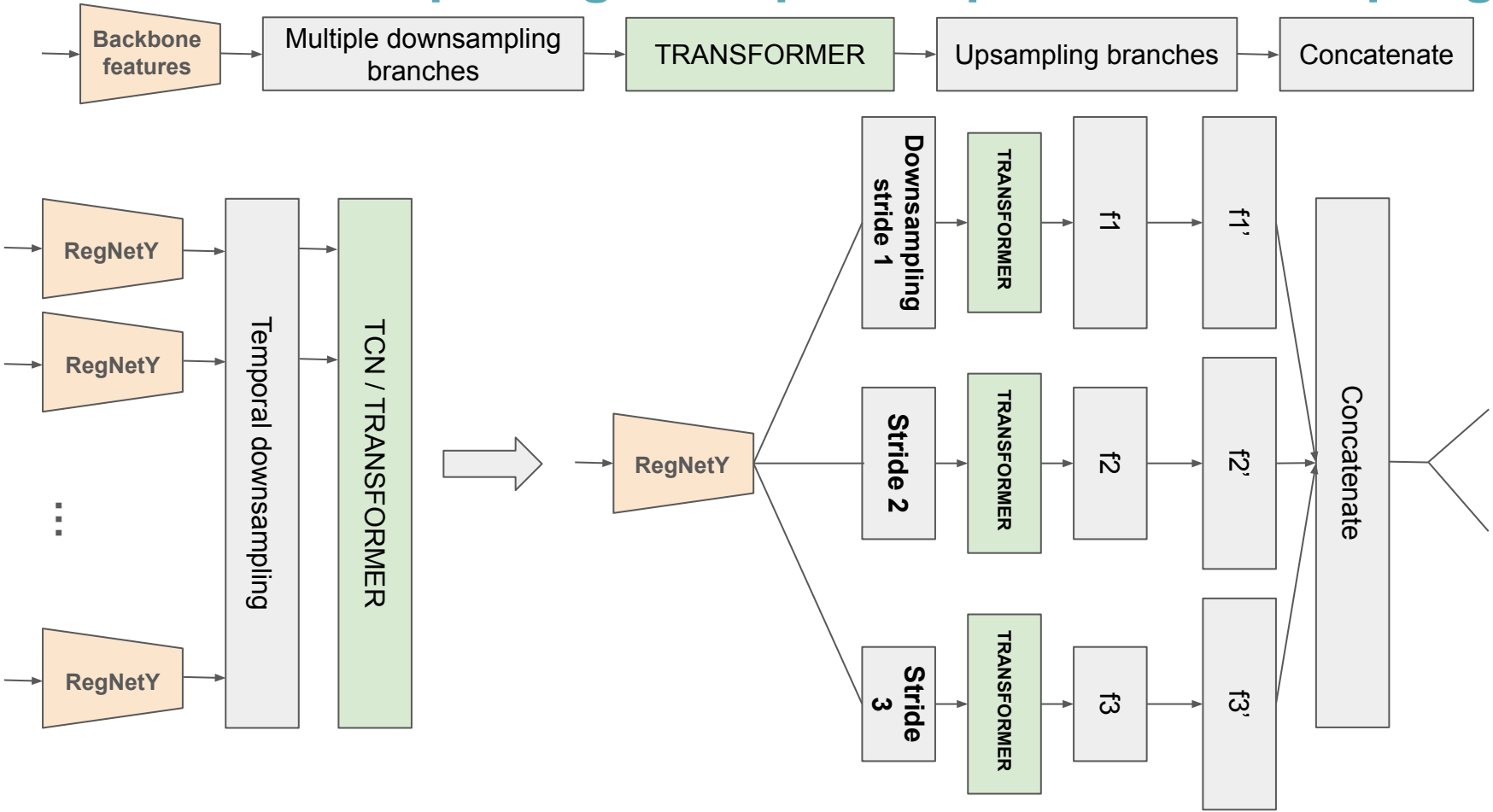
Frame	Classification	
	Class 0	Class 1
0	0.1	0.9
1	0	0
2	0	0
3	0.6	0.4
4	0.2	0.8
5	0	0

Prediction 2

- i=2
- Position = 2×2=4
- Offset = -0.5
- Corrected = round(4−0.5)=4

→ Place [0.2, 0.8] at frame 4.

1.b. Ball action spotting: Multiple temporal downsampling



1.c. Ball action spotting: Experiments (I)

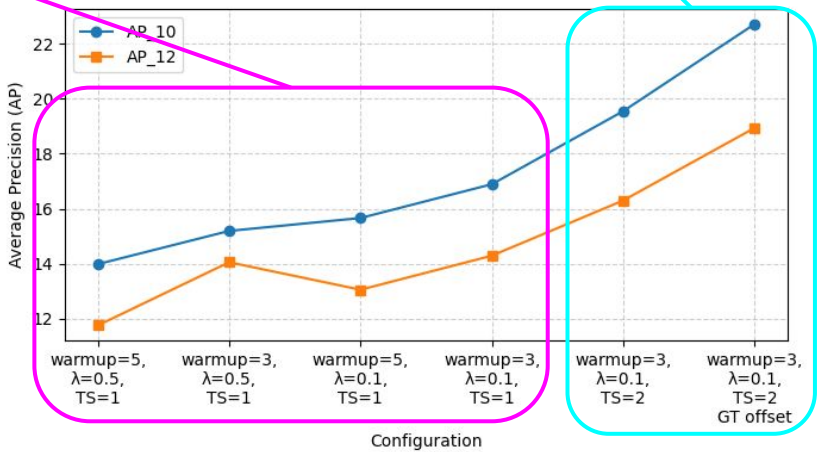
Fixed Backbone: regnety 002

We started with last week's **TCN** and adding **temporal downsample** 1, i.e., only changing **loss function**.

Obs: **less warm-up** is better because we train only for 20 epochs.

Obs: **lower lambda** (less weight to offset head) yields better results.

Next: Temporal downsampling. First with original setup. Then, the same experiment is repeated with a **modified GT offset** creation (as explained earlier), using a fixed max distance of 8.



Epochs: 20.
LR: 0.0001.
Frames: 50.
Stride: 2.
Temporal model: TCN.

CHANGES:
- Warm-up: 5 and 3
- Lambda: 0.5 and 0.1
- Temporal Stride: 1 and 2

Best results:
warm-up = 3 and λ = 0.1 with created GT offset.

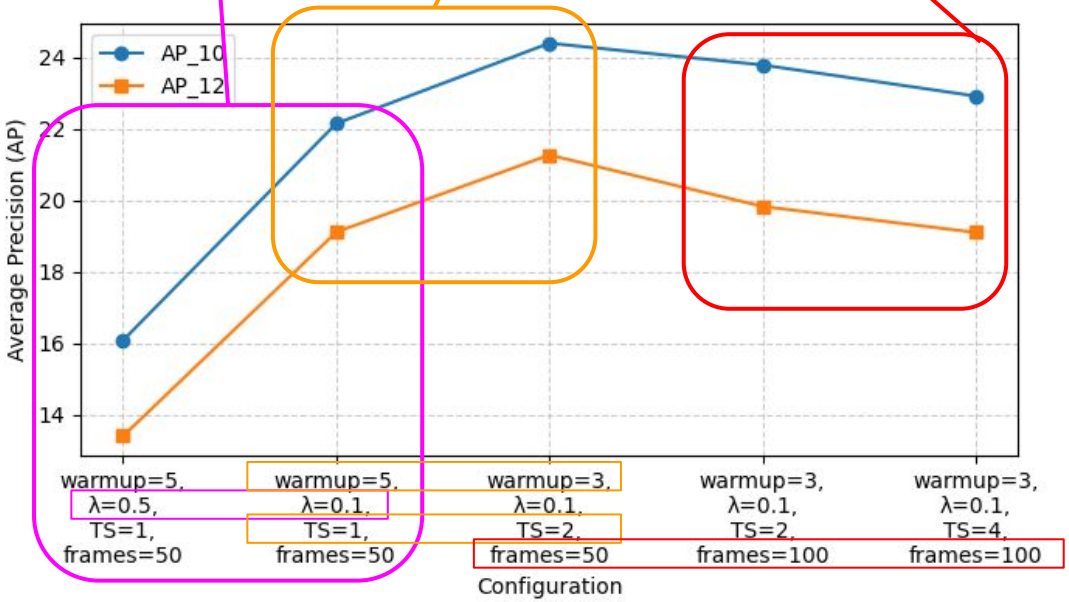
1.c. Ball action spotting: Experiments (II)

Now with transformers!

Now we use a **transformer** instead of a TCN.

First changing λ: again, **less weight in offset head** loss gives better results ✅.

More changes:
- Reducing warm-up ✅
- Increase temporal stride ✅, but not too much ❌
- Increase number of frames ❌



Epochs: 20.
LR: 0.0001.
Stride: 2.
Temporal model: Transformer.

CHANGES:
- Warm-up: 5 and 3
- Lambda: 0.5 and 0.1
- Temporal Stride: 1 and 2
- Frames: 50 and 100

Best results:
warm-up = 3, λ = 0.1, 50 frames and temporal offset of 2.

1.c. Ball action spotting: Experiments (III)

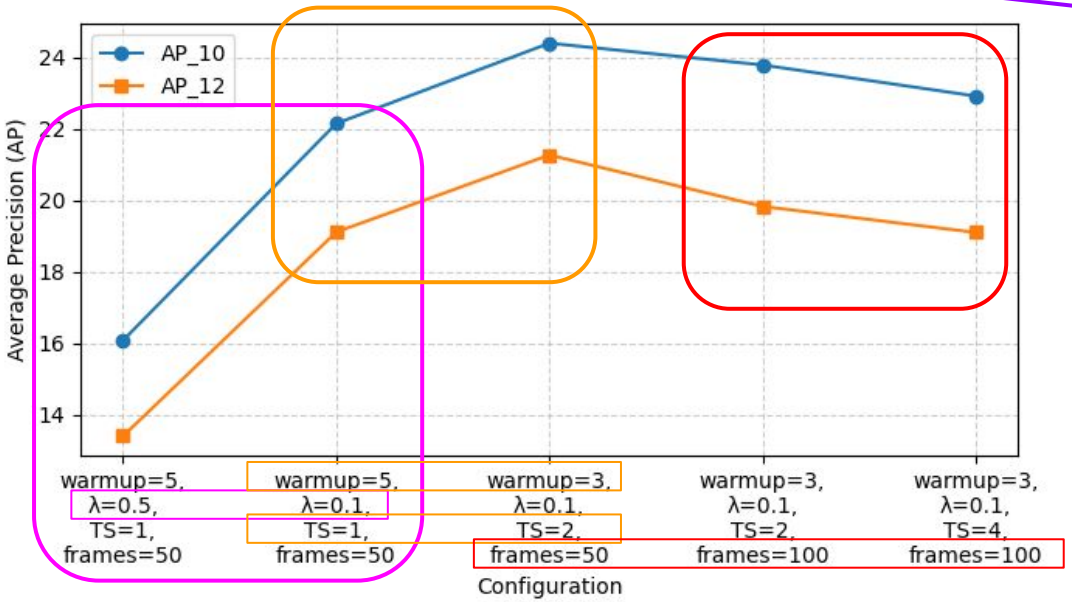
Now with transformers!

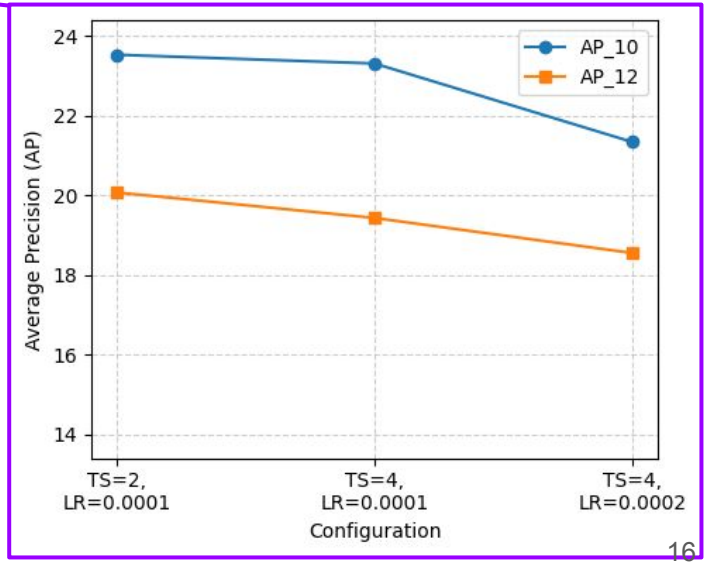
Now we try with **100 frames** and with a starting stride of 3 for the clip. We also tried increasing LR but still didn't perform better.

Epochs: 20.
Warm-up: 3.
Frames: 100.
Stride: 3.
Temporal model: Transformer.
λ: 0.1

CHANGES:
- Temporal Stride: 2 and 4
- Learning Rate

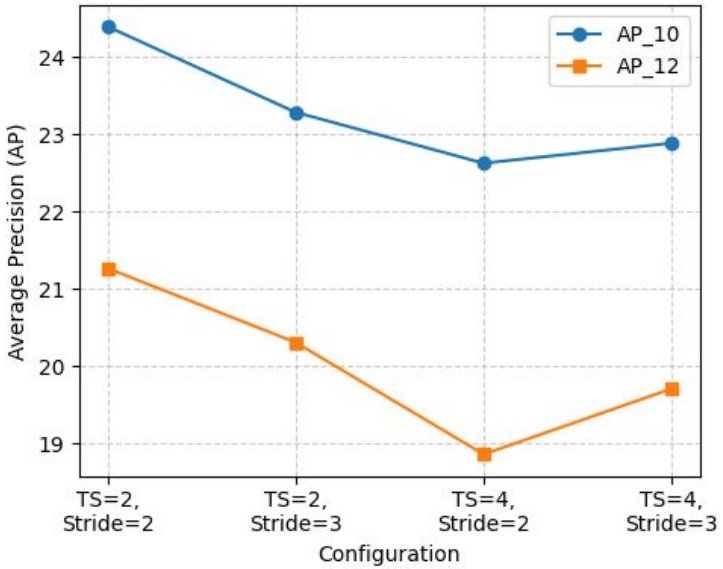
Best results:
Better than previous (red) with 100 frames, but still not best (orange).





1.c. Ball action spotting: Experiments (IV)

Since increasing number of frames didn't improve results, we go back to 50. We test different combinations of **initial stride** (to create the clip) and **internal downsampling** to see their impact on model performance.



Epochs: 20.
LR: 0.0001.
Frames: 50.
Warm-up: 3.
Temporal model: Transformer.
Lambda: 0.1

CHANGES:
- Stride (clip): 2 and 3
- Temporal Stride: 2 and 4

Best results:
Still obtained using stride 2 and downsampling 2. Increasing either one hurts performance with our current setup. Maybe it could work better by tuning other hyper-parameters, but we didn't have time to test.

1.c. Ball action spotting: Experiments (V)

Finally, we tested the architecture with **multiple downsampling resolution**. We used a **1-2-4 stride** to create three branches with the backbone features.

First, we use the best parameters obtained using this architecture. Then, we run a test by increasing the lambda value of the loss function and the learning rate of the optimizer.

Epochs: 20
LR: 0.0001
Clip_stride: 2
Frames: 50.
Warm-up: 3
Temporal model: Transformer.
Lambda: 0.1
Downsampling_stride: 1-2-4

mAP@12: 24.76
mAP@10: 21.22

Epochs: 20
LR: 0.0001
Clip_stride: 2
Frames: 50.
Warm-up: 3.
Temporal model: Transformer.
Lambda: 0.3
Downsampling_stride: 1-2-4

mAP@12: 18.03
mAP@10: 20.22

Epochs: 20
LR: 0.00005
Clip_stride: 2
Frames: 50
Warm-up: 3
Temporal model: Transformer
Lambda: 0.1
Downsampling_stride: 1-2-4

mAP@12:
mAP@10:

1.c. Ball action spotting: Final results

Best/final configuration:

Epochs: 20
Warm-up: 3
LR: 0.0001
Frames: 50
Stride: 2
Temporal Stride: 2
Temporal Model:Transformer
Lambda: 0.1

Class	This Week	Last Week
pass	56.4	53.6
drive	47.23	38.99
header	21.9	18.16
high pass	50.7	38.11
out	8.5	2.04
cross	28.34	21.24
throw in	17.21	5.16
shot	11.64	5.02
ball player block	2.14	1.14
player successful tackle	0	0
free kick	11.11	0
goal	0	0
mAP@12	**21.27**	**15.29**
mAP@10	**24.40**	**18.34**

2. Conclusions

- Temporal downsampling improves results if we use a displacement head
- Transformer architecture outperforms TCN architecture
- Increasing the stride above 2 to clear the clips has worsened the results
- Increasing the number of frames worsens the results.
- The best temporal downsampling is with stride 2. Increasing this value worsens the results
- The construction of labels for the offset head during training affects the result
- The way the loss from the two heads is combined impacts the model's performance. It must be weighted appropriately
- The architecture with multiple downsampling resolution has not improved over the simpler one, probably due to a lack of parameter tuning
- There are a large number of parameters and combinations to optimize and all of them affect the performance of the model in some way