SoGAR: Self-Supervised Spatio-Temporal Attention-Based GAR

Mohammadmahdi Zare'i

October 5, 2025

Motivation

- Understanding collective human behavior from video is essential for sports, surveillance, and social scene analysis.
- Traditional methods rely on supervised labels; SoGAR introduces a **self-supervised** learning approach.
- It learns from unlabeled video by aligning local and global temporal representations.

SoGAR Overview

- **Goal:** Learn spatio-temporal video representations without labels.
- **Architecture:** ViT-Base backbone with divided space-time attention.
- **Training:** Teacher-student EMA setup with TCL and SCL losses.
- **Outcome:** A pretrained model transferable to labeled group activity datasets.

Input Sampling

- Global view: K_g frames, high resolution (480x480).
- Local views: multiple K_l-frame clips (e.g., 2–16), low resolution (96x96).
- Bounding boxes guide local crops when available.
- $q = 16$ local views per video for self-supervised comparison.

Patch Embedding

- Each frame is divided into 16×16 patches.
- Linear projection maps patches to 768-dimensional embeddings.
- A learnable [CLS] token represents the entire clip.

$$x_p = W_p \cdot \text{Flatten}(patch), \quad x = [CLS; x_{p1}, \ldots, x_{pN}]$$

Divided Space–Time Attention Blocks

- 12 Transformer blocks (ViT-Base).
- Each block splits attention into spatial and temporal sub-layers.
- Spatial attention captures within-frame relations.
- Temporal attention models motion and group dynamics.

$$x' = \mathrm{MHA} * \mathit{space}(x) \; x'' \qquad = \mathrm{MHA} * \mathit{time}(x')$$

Projection Head

- Two-layer MLP: $768 \to 4096 \to 256$.
- BatchNorm and ReLU between layers.
- Outputs normalized feature vector z.

```
proj = nn.Sequential(
nn.Linear(768, 4096),
nn.BatchNorm1d(4096),
nn.ReLU(),
nn.Linear(4096, 256)
)
z = F.normalize(proj(cls), dim=-1)
```

Predictor Head (Student Only)

- Predicts teacher embedding from student projection.
- Two-layer MLP with identical structure to projector.
- Encourages temporal alignment via prediction consistency.

```
predictor = nn.Sequential(
nn.Linear(256, 4096),
nn.BatchNorm1d(4096),
nn.ReLU(),
nn.Linear(4096, 256)
)
p_l = predictor(z_l)
```

Teacher–Student Setup (EMA Update)

- Teacher shares architecture with student but updated via EMA.
- Prevents collapse and provides a stable learning target.

$\theta_t \leftarrow m, \theta_t + (1 - m), \theta_s$

```
with torch.no_grad():
    for pt, ps in zip(teacher.parameters(), student.parameters()):
        pt.data = pt.data * m + ps.data * (1 - m)
```

Temporal Collaborative Learning (TCL) Loss

- Aligns student local prediction p_l with teacher global embedding z_g.
- Uses cross-entropy over normalized vectors.

$$L_{TCL} = -z_g \cdot \log(p_l)$$

```
loss_tcl = - (z_g.detach() * torch.log_softmax(p_l, dim=-1)).sum(
    dim=-1).mean()
```

Spatio-Temporal Cooperative Learning (SCL) Loss

- Aligns global teacher embedding z_g with all local predictions p_l^i.
- Enforces consistency across multiple local views.

$$L_{SCL} = \frac{1}{q} \sum_{i=1}^{q} -z_g \cdot \log(p_l^i)$$

```
loss_scl = sum(
- (z_g.detach() * torch.log_softmax(p, dim=-1)).sum(dim=-1).mean()
for p in p_locals
) / len(p_locals)
```

Augmentations and View Sampling

- Color jitter ($p = 0.8$), grayscale ($p = 0.2$) on all views.
- Gaussian blur ($p = 0.1$) and solarization ($p = 0.2$) for global views only.
- Encourages invariance to appearance changes.

```
transform = Compose([
ColorJitter(p=0.8),
RandomGrayscale(p=0.2),
RandomApply([GaussianBlur()], p=0.1),
RandomApply([Solarize()], p=0.2)
])
```

Training Pipeline (End-to-End)

- Global view processed by teacher $\rightarrow z_g$.
- Local views processed by student $\rightarrow z_l \rightarrow p_l$.
- Total loss: $L = L_{TCL} + L_{SCL}$.
- Teacher updated via EMA after each step.

```
loss = loss_tcl(z_g, p_l) + loss_scl(z_g, p_locals)
optimizer.zero_grad()
loss.backward()
optimizer.step()
update_ema(student, teacher)
```

Results and Outcomes

- Self-supervised training achieves strong performance on Volleyball, NBA, and JRDB-PAR datasets.
- Learns rich spatio-temporal representations transferable to multiple downstream tasks.
- Efficient ViT-Base backbone with scalable training via view sampling.