BASAR: Black-box Attack on Skeletal Action Recognition

Supplemental Document

Yunfeng Diao\textsuperscript{1,2}*, Tianjia Shao\textsuperscript{3†}, Yong-Liang Yang\textsuperscript{4}, Kun Zhou\textsuperscript{3}, He Wang\textsuperscript{1}
\textsuperscript{1}University of Leeds, UK \textsuperscript{2}Southwest Jiao tong University, China
\textsuperscript{3}State Key Lab of CAD&CG, Zhejiang University, China \textsuperscript{4}University of Bath, UK
dyf@my.swjtu.edu.cn, tjshao@zju.edu.cn, y.yang@cs.bath.ac.uk, kunzhou@zju.edu.cn, h.e.wang@leeds.ac.uk

1. Additional Experimental Details

1.1. Implement Details and Experimental Settings

We first give details about the Random Exploration and Aimed Probing. For easy reference, the random exploration is reformulated in Eq. 1:

\[
\tilde{x} = x' + W \Delta, \\
\text{where } \Delta_s = R_s - (R_s^T d_s) d_s, \quad d_s = \frac{x_s - x'_s}{\|x_s - x'_s\|}, \\
R_s = \lambda \frac{r}{\|r\|} \|x_s - x'_s\|, \quad r \in N(0, I),
\]

where \(\tilde{x}\) is the new perturbed sample, \(x\) and \(x'\) are the attacked motion and current adversarial sample. We use joint positions and the subscript \(s\) indicates either the \(x\), \(y\), or \(z\) joint coordinate. The update on \(x'\) is \(\Delta\) weighted by \(W\) - a diagonal matrix with joint weights. \(\Delta_s\) controls the direction and magnitude of the update, and depends on two variables \(R_s\) and \(d_s\). \(d_s\) is the directional vector from \(x'\) to \(x\). \(R_s\) is a random diagonal vector sampled from a Normal distribution \(N(0, I)\) where \(I\) is an identity matrix, \(I \in R^{2 \times 2}\), \(z = mn/3, m\) is the number of DoFs in one frame and \(n\) is the total frame number. This directional vector is scaled by \(\|x_s - x'_s\|\) and \(\lambda\).

The aimed probing is reformulated by Eq. 2:

\[
\tilde{x} = x' + \beta(x - x'),
\]

where \(\beta\) is a forward step size that can also be dynamically adjusted. \(\beta\) is decreased by half to conduct the aimed probing again if \(\tilde{x}\) is not adversarial; otherwise, \(\beta\) is doubled, then we enter the next sub-routine.

In random exploration, we aim to find an adversarial sample that is closer to \(x\). However, as the shape of the local space is unknown and highly nonlinear, we do sampling to exploit it. Therefore, we execute multiple random explorations instead of only one to get \(q\) intermediate results in a sub-routine call, and compute the attack success rate. If the rate is less than 40%, \(\lambda\) is reduced by 10% as it means that we are very close to the classification boundary \(\partial C\) and \(\lambda\) is too big; if it is higher than 60%, \(\lambda\) is increased by 10%; otherwise, we do not update \(\lambda\).

For targeted attack, we randomly select one adversarial sample from the \(q\) intermediate samples to do aimed probing. This is mainly to ensure that the direction of the aimed probing is random. Although multiple samples can be selected, it would incur more computational costs with little gain shown by our preliminary experiments. For untargeted attack, the \(q\) results are normally in different classes which we call adversarial classes. The attack difficulty varies depending on the choice of samples. Usually the closer the adversarial sample is to the original sample, the easier the attack. Therefore, we randomly select one sample in each adversarial class to conduct aimed probing, then only keep the one that has the smallest distance to the original motion \(x\) after the aimed probing. In the end, when the adversarial sample is near to the original motion, we set a threshold value \(\tau\) to ensure that \(\lambda\) is not higher than \(\tau\). This is to ensure that the attack can eventually converge.

In all experiments, we set \(q = 5\). The initial \(\beta\) is set to 0.95. The initial \(\lambda\) is set to 0.2 when attacking SGN model and 0.1 on both STGCN and MSG3D. \(\tau\) is set to 1.5 on SGN and 0.4 on both STGCN and MSG3D. We set the spinal joint weights to 0 in \(W\) and other joint weights to 1. For untargeted attack, we set \(\epsilon = 0.1\) on both HDM05 and NTU, 0.05 on Kinetics. For targeted attack, \(\epsilon\) is set to 0.5 on HDM05 and both 0.2 on NTU and Kinetics. Considering the optimization speed, it is unrealistic to execute the manifold projection in every iteration. We therefore execute it every 100 iterations on HDM05 and every 250 iterations on NTU and Kinetics.

The adversarial samples are computed using PyTorch on a PC with a NVIDIA GTX 2080Ti GPU and a Xeon Silver 4216 CPU. We also show how different metrics vary based on the number of actual queries BASAR makes to the at-
tacked model. The evaluation versus number of queries are shown from Fig 1 to Fig 3. Being consistent with our analysis in the paper, compared with STGCN and MSG3D, SGN usually converges faster but it is difficult for BASAR to further improve the adversarial sample as it does on STGCN and MSG3D. We speculate that this is due to the semantic information that SGN uses prevents small perturbations from altering the class labels.

| Models | HDM05 Queries | HDM05 Time | NTU Queries | NTU Time | Kinetics Queries | Kinetics Time |
|--------|---------------|------------|-------------|----------|-----------------|--------------|
| STGCN  | UA 3636       | 4          | UA 7337     | 12       | UA 7167         | 28           |
|        | TA 8862       | 15         | TA 15724    | 16       | TA 15234        | 41           |
| MSG3D  | UA 3722       | 6          | UA 14640    | 18       | UA 7190         | 29           |
|        | TA 9111       | 16         | TA 23227    | 30       | TA 15416        | 56           |
| SGN    | UA 974        | 4          | UA 623      | 5        | UA 228          | 10           |
|        | TA 277        | 3          | TA 260      | 4        | TA 180          | 8            |

Table 1. The averaged number of queries and consuming time(min) for generating an adversarial sample on different models and datasets.

1.2. Detailed Perceptual Studies

In all 50 subjects (age between 18 and 54), 86% of users are aged under 30 and 88% are male. The users have various background. Around 25% users have research expertise in human activity recognition or adversarial attack; another 20% have general deep learning or computer vision background; 45% people study in engineering (e.g. mechanical, electrical and control). The other users have different arts background. By comparing their performance we found that the age, gender and work/research background actually do not have obvious influence to the results. The results are mainly dependent on the quality of the adversarial samples.

**Deceitfulness.** This study is to test: whether BASAR visually changes the meaning of the motion and whether the meaning of the original motion is clear to the subjects. In each user study, we randomly choose 45 motions (15 from STGCN, MSG3D and SGN respectively) with the ground truth label and after-attack label for 45 trials. In each trial, the video is played for 6 seconds then the user is asked the question, ‘which label best describes the motion? and choose Left or Right’, with no time limits.

**Naturalness.** This ablation study is to test whether on-manifold adversarial samples look more natural than off-manifold samples. In this user study, we perform ablation studies to test whether on-manifold adversarial samples look more natural than off-manifold samples. We design two settings: MP and No MP. MP refers to BASAR, with manifold projection. No MP is where the proposed method without manifold projection. In each study, 60 (20 from STGCN, MSG3D and SGN respectively) pairs of motions are randomly selected for 60 trials. Each trial includes one from MP and one from No MP. The two motions are played together for 6 seconds twice, then the user is asked, ‘which motion looks more natural? and choose Left, Right or Can’t tell’, with no time limits.

**Indistinguishability.** Indistinguishability is the strictest test to see whether adversarial samples by BASAR can survive a side-by-side scrutiny. In each user study, 40 pairs of motion are randomly selected, half from STGCN and half from MSG3D. For each trial, two motions are displayed side by side. The left motion is always the original and the user is told so. The right one can be original (sensitivity) or attacked (perceivability). The two motions are played together for 6 seconds twice, then the user is asked, ‘Do
they look same? and choose Yes or No', with no time lim-
its. This user study serves two purposes. Perceivability is
a direct test on indistinguishability while sensitivity aims
to screen out users who tend to choose randomly. We dis-
card any user data which falls below 80% accuracy on the
sensitivity test.

2. Visual Results and Confusion Matrices

The visual results on various datasets and models are
shown from Fig. 4 to Fig. 9. As we can see, the adversarial
samples on STGCN and MSG3D in general are very hard
to be distinguished from the attacked motion. The results
on SGN have the same semantic meanings and are almost
equally hard to be distinguished from the original motion
in untargeted attack. However, when it is targeted attack
and the target label is very different from the original la-
bel, BASAR sometimes generate adversarial samples with
visible differences. We show some failures here (Fig. 6 Bot-
tom and Fig. 9 Bottom.). These adversarial samples might
survive a visual examination if shown alone but might not
be able to survive a side-by-side comparison with the original
motions in our rigorous perceptual studies. This is also
consistent with our numerical evaluation.

In NTU, there are actions containing a single person or
two persons. We, therefore, attack them separately. In tar-
gected attack, if the attacked motion is a single-person ac-
tion, the target class is also a single-person action where we
randomly select a motion to initiate the attack. Similarly,
if the attacked motion is a two-person action, we select a
two-person motion. In untargeted attack, we do not need to
initiate the attack separately and can rely on BASAR to find
the adversarial sample that is closest to the original motion.
The confusion matrices across various datasets and models
are shown from Fig. 10 to Fig. 15.

In untargeted attack, we find that random attacks easily
converge to a few action classes in a dataset. We call them
high-connectivity classes. For example, actions on STGCN
tend to be attacked into ‘Jump Jack’ (number 20) and ‘Kick
left front’ (21) on HDM05, and into ‘Use a fan’ (48) on
NTU, regardless how they are initialized; Similarly, ac-
tions on MSG3D tend to be attacked into ‘Cartwheel’ (0)
and ‘Kick right front’ (23) on HDM05, and into ‘Use a
fan’ (48) on NTU; actions on SGN tend to be attacked into
‘Cartwheel’ (0) and ‘Jump Jack’ (20) on HDM05, and into
‘Hopping’ (25) on NTU. The theoretical reason is hard to
identify but we have the following speculations. Since un-
targeted attack starts from random motions, it is more likely
to find the adversarial sample that is very close to the origi-
nal motion on the classification boundary. Usually this
adversarial sample is in a class that shares the boundary
with the class of the original motion. It is possible that
these high-connectivity classes share boundaries with many
classes so that random attacks are more likely to land in
these classes. In addition, the connectivity of classes heav-
ily depends on the classifier itself and that is why different
classifiers have different high-connectivity classes. In tar-
gected attack, since our target labels are randomly selected,
the confusion is more uniformly distributed, covering all
classes.

Figure 4. STGCN on HDM05. The ground truth label ‘Rotate right
arms backward’ is misclassified as ‘Clap above hand’ on untar-
gected attack, and ‘Kick left side’ on targeted attack.

Figure 5. MSG3D on HDM05. The ground truth label ‘Elbow
to knee’ is misclassified as ‘Sit down’ on untargeted attack, and
‘Cartwheel’ on targeted attack.

Figure 6. SGN on HDM05. The ground truth label ‘Punch right
front’ is misclassified as ‘Punch right side’ on untargeted attack, and
‘Standing and throw down’ on targeted attack.
Figure 7. STGCN on NTU. The ground truth label ‘Taking a selfie’ is misclassified as ‘Stand up’ on untargeted attack, and ‘Hand waving’ on targeted attack.

Figure 8. MSG3D on NTU. The ground truth label ‘Rub two hands together’ is misclassified as ‘Clapping’ on untargeted attack, and ‘Pick up’ on targeted attack.

Figure 9. SGN on NTU. The ground truth label ‘Take off glasses’ is misclassified as ‘Wear on glasses’ on untargeted attack, and ‘Wipe face’ on targeted attack.

Figure 10. Confusion matrix of STGCN on HDM05. Left is untargeted attack and right is targeted attack. The darker the cell, the higher the value.

Figure 11. Confusion matrix of MSG3D on HDM05. The left one is untargeted attack and right is targeted attack. The darker the cell, the higher the value.

Figure 12. Confusion matrix of SGN on HDM05. The left one is untargeted attack and right is targeted attack. The darker the cell, the higher the value.
3. Additional Details of Manifold Projection

The original problem is as follows:

\[
\begin{aligned}
\text{minimize} & \quad L(\theta, \theta') + w L(\ddot{\theta}, \ddot{\theta'}) \\
\text{subject to} & \quad \theta_i^{\min} \leq \theta_i' \leq \theta_i^{\max}, \\
& \quad C_{x'} = c \text{ (targeted)} \text{ or } C_{x'} \neq C_x \text{ (untargeted)}. \quad (3)
\end{aligned}
\]

by \( \theta_i^{\min} \) and \( \theta_i^{\max} \). \( \ddot{\theta} \) and \( \ddot{\theta}' \) are the 2nd-order derivatives of \( \theta \) and \( \theta' \). \( w \) is a weight. \( L \) is the Euclidean distance. Such a nonlinear optimization problem can be transformed to a barrier problem [1]:

\[
\begin{aligned}
\min_{\theta'} & \quad L(\theta, \theta') + w_1 L(\ddot{\theta}, \ddot{\theta'}) + \sum_{i} O \mu_i \ln (\theta_i' - \theta_i^{\min}) \\
& + \sum_{i} O \mu_i \ln (\theta_i^{\max} - \theta_i') \quad (4)
\end{aligned}
\]

where \( \mu_i \) is a barrier parameter. \( O \) is the total number of joints in a skeleton. For notational simplicity, we notate \( f(\theta') = L(\theta, \theta') + w L(\ddot{\theta}, \ddot{\theta}') \). The Karush-Kuhn-Tucker conditions [2] for the barrier problem Eq.5 can be written as:

\[
\begin{aligned}
\nabla f(\theta') + \sum_{i} O \frac{\mu_i}{\theta_i' - \theta_i^{\min}} - \sum_{i} O \frac{\mu_i}{\theta_i^{\max} - \theta_i'} = 0 \\
\mu_i \geq 0, \text{ for } i = 1, \ldots, O \\
\sum_{i} O \mu_i \ln (\theta_i' - \theta_i^{\min}) = 0 \\
\sum_{i} O \mu_i \ln (\theta_i^{\max} - \theta_i') = 0 \quad (5)
\end{aligned}
\]

We apply a damped Newton’s method [4] to compute an approximate solution to Eq. 5. More implementation details about the primal-dual interior-point method can be found in [3].

References

[1] Andrew R. Conn, Nicholas I. M. Gould, Dominique Orban, and Philippe L. Toint. A primal-dual trust-region algorithm for non-convex nonlinear programming. Math. Program., 87(2):215–249, 2000. 5

[2] Harold W Kuhn and Albert W Tucker. Nonlinear programming. In Traces and emergence of nonlinear programming, pages 247–258. Springer, 2014. 5

[3] Andreas Wächter and Lorenz T Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Mathematical programming, 106(1):25–57, 2006. 5

[4] Tjalling J Ypma. Historical development of the newton–raphson method. SIAM review, 37(4):531–551, 1995. 5