Domain Knowledge-Informed Self-Supervised Representations for Workout Form Assessment

Paritosh Parmar[1,2], Amol Gharat[2], and Helge Rhodin[1]

[1] University of British Columbia
[2] FlexAI Inc.

1 Description of Exercise Errors

We provide a description of various errors from various exercises in our dataset in Table 1. It lists all the exercises, all the types of errors, and for each error, we have provided illustrations of correct and incorrect forms.

2 Additional Discussion on Approaches

For our CVCSPC approach, we believe that during self-supervised training, the network learns pose sensitive features from rough/noisy pose matching-contrasting. A more refined mapping of these features to error labels is learned during the fine-tuning phase.
For MD approach, through contrastive learning, the 3DCNN learns to capture the local, anomalous motions that are accentuated in our specially created triplets. During the finetuning phase, the learnt representations are calibrated so as to distinguish between harmful irregularities and harmless variations.

3 Implementation Details

We implemented all of our models using PyTorch [4]. In the following, we provide further implementation details regarding finetuning phase:

CVCSPC/PAD: During finetuning phase, we use 180 frames from an exercise sequence. To train the ResNet-1D temporal model [3], we optimize using ADAM optimizer [2] for 50 epochs with an initial learning rate of 1e-4 and with a batchsize of 25.

MD: We use 32 frames from an exercise sequence. We add fully-connected layers for classification on top of the backbone. We optimize using ADAM optimizer for 20 epochs with a batchsize of 5.

TDM: When evaluating the static errors, we passed TDMs through multiple fully-connected layers followed by ReLU non-linearity to do the classification. For this, we experimented with using one to five fully-connected layers, and found that four fully-connected layers worked the best.

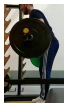
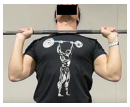
Exercise	Error type	Correct	Incorrect
BackSquat	Knees Inward Error	Knees pointing outwards 	knees buckling in 
BackSquat	Knees Forward Error	Knees should be aligned over the toes 	knees moving forward excessively 
BackSquat	Shallow Squat Error	Glutes below knees-line 	glutes above knees-line 
BarbellRow	Lumbar Error	Lower back should be neutral 	Exaggerated curvature of the lumbar spine 
BarbellRow	Torso-Angle Error	Back aligned with the hips at a 45°-90° angle with the core 	Too high less than 45° torso inclination / Too low, torso below parallel, compared to the floor 
OverheadPress	Elbow Error	Elbows underneath the wrists 	Elbows flaring out or caving in 
OverheadPress	Knees Error	Knees should be locked at all times 	Knees are bent. Knees go from being bent to straight to help jerk the bar up (because the weights overly heavy for the person) 

Table 1: **Description of various errors** covered in our dataset.

BarbellRow splits: For BarbellRow exercise, we used a comparatively small training set, in particular, we used train/val/test set of sizes: 2075/2227/2191.

Further Details Regarding Pose and Appearance Disentangling Baseline. In this baseline, we aim to disentangle pose of the humans doing exercises from their appearance (in computer vision sense). To accomplish this, we leverage an autoencoder setup as show in Fig. 1. This approach involves two randomly selected frames, X, and Y, from an exercise video instance and comprises of the following four stages:

Image encoding: We pass the input frames through encoder, E, and obtain the output of the last layer convolutional block, *e.g.*, the last bottleneck of Resnet-18.

Feature partition: We partition the feature volume obtained in the previous step at the certain along the channel dimension. We take inspiration from pose estimation networks, where each body joint is spatially represented in a particular feature plane. Typically, the number of joints are in the range of 18-24. We arbitrarily assume the number of joints to be 21, and therefore, partition the feature volume in a way that we set 21 feature maps for representing pose information, while the remaining for appearance information. The partitioned features are then projected into a lower dimensional feature space using a fully-connected layer. Appearance and pose corresponding to an image, x, are denoted by $\psi(x)$, and $\pi(x)$, respectively. Appearance and pose features concatenated together form the overall image representation, ϕ. We take care to keep the dimensionality of pose feature vector to be very compact, so as to not allow any appearance information to leak through it. Using appearance vector of much larger size than the pose vector also helps in ensuring that appearance features do not collapse to trivial/null solution. Finally, Pose vector, P (in Fig. 1), is only 32-dim vector. We then use the pose features for error detection.

Feature swapping: After computing pose and appearance features for both frames, we swap the appearance feature vectors before passing through the decoder (discussed in the next stage). Before swapping operation, the overall image representation would be,

$$\phi(X) = \psi(X) \oplus \pi(X), \tag{1}$$
$$\phi(Y) = \psi(Y) \oplus \pi(Y), \tag{2}$$

where, $\oplus$ denotes concatenation operation. After swapping operation, the overall image representation would be,

$$\phi(X) = \psi(Y) \oplus \pi(X), \tag{3}$$
$$\phi(Y) = \psi(X) \oplus \pi(Y). \tag{4}$$

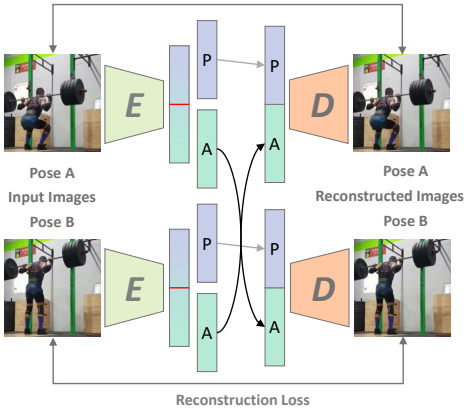


Fig. 1: **Pose and appearance disentangling** via feature partition and swapping. E and D denote encoder and decoder CNNs; A and P denote appearance and pose feature representations.

Image reconstruction: After swapping step, we pass the overall image representations through image decoder/reconstruction network, D, to reconstruct the input frames. Reconstruction loss is a sum of pixelwise (L1 and L2 losses), and perceptual loss [1] as in Eq. 8.

$$\mathcal{L}_{L1} = \frac{1}{i.j} \sum X_{i,j} - \widetilde{X}_{i,j} \tag{5}$$

$$\mathcal{L}_{L2} = \frac{1}{i.j} \sum ||X_{i,j} - \widetilde{X}_{i,j}||_2 \tag{6}$$

$$\mathcal{L}_{perc} = \frac{1}{m} \sum ||G(X)_{BBox} - G(\widetilde{X})_{BBox}||_2 \tag{7}$$

$$\mathcal{L}_{recon} = \lambda_1 \mathcal{L}_{L1} + \lambda_2 \mathcal{L}_{L2} + \lambda_3 \mathcal{L}_{perc} \tag{8}$$

X and $\widetilde{X}$ represent original and the reconstructed images. G represents layer VGG16 network [6] upto `relu2_2`, trained on ImageNet [5]. For computing losses for this approach, we use bounding box around the person and consider only the image/feature area within the box when computing losses in equations 5 to 8.

4 Additional Information on Exercise Errors

The errors we target in this work include those where an excessive amount of strain on certain ligaments/tendons/muscles, which are likely to cause injuries and those where the wrong muscle group is activated due to the subject not maintaining the proper form. For example, moving the knees too much forward during the squats, places a huge strain on the patellar tendon and anterior cruciate ligament (ACL), and calling on the quadriceps to do the most work

during the squat instead of the glutes and hamstrings sharing the load. These kinds of errors also result in lower than optimal gains.

5 Attention Visualization

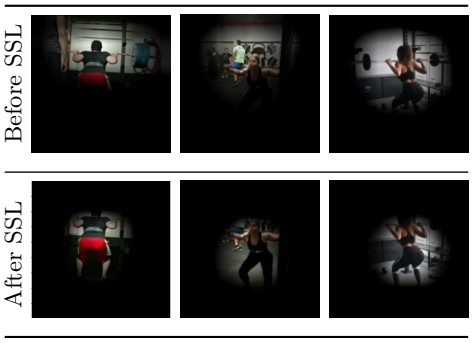


Fig. 2: **Attention visualization.**

We compared the location of attentions before and after our self-supervised training (refer to Fig. 2). We found that after self-supervised training, the CNN focused more on the human doing the exercise, around the important body parts, for example, legs in case of squats.

References

1. Johnson, J., Alahi, A., Fei-Fei, L.: Perceptual losses for real-time style transfer and super-resolution. In: European conference on computer vision. pp. 694–711. Springer (2016) 4
2. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014) 1
3. Ogata, R., Simo-Serra, E., Iizuka, S., Ishikawa, H.: Temporal distance matrices for squat classification. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. pp. 0–0 (2019) 1
4. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. Advances in neural information processing systems **32**, 8026–8037 (2019) 1
5. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: Imagenet large scale visual recognition challenge. International journal of computer vision **115**(3), 211–252 (2015) 4
6. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014) 4