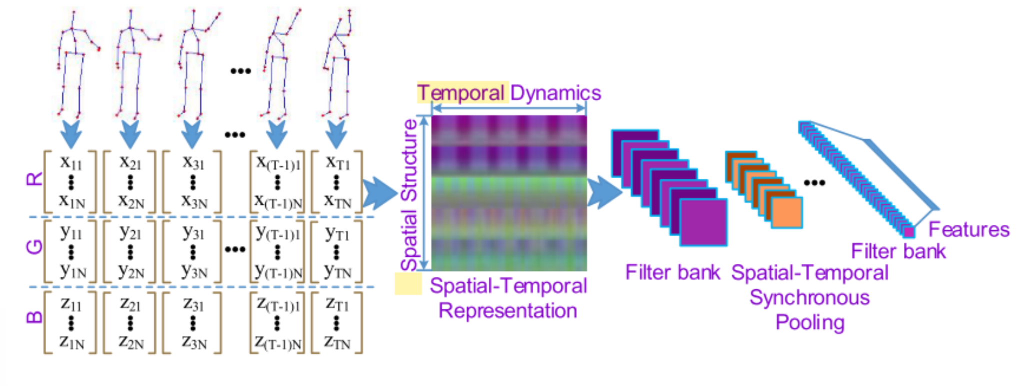
Protocol 18: 2D CNN Training

10.05.2019

Convolutional Neural Networks (CNN)

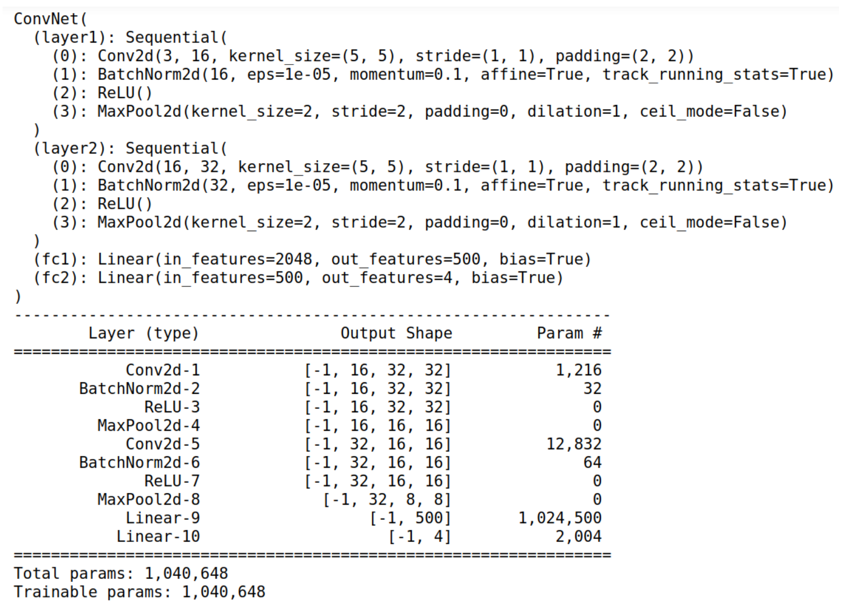
In addition to the ST-GCN (Spatial Temporal Graph Convolutional Networks), which represents the skeleton sequence as a graph, some studies convert the skeleton sequence into 2d images and use 2D CNNs for action recognition [1, 2, 3]. The idea is that the skeletal joints from each video frame are concatenated as a vector, and the vectors from different frames are combined as an 2D array or an image, as shown in the left side of Figure 1. The dimension of the joints represents the number of channels of the image.



R
G
B
Temporal Dynamics
Spatial Structure
Spatial-Temporal Representation
Filter bank
Spatial-Temporal Synchronous Pooling
Filter bank
Features

Figure 1: Skeletal joints represented as an image and the training process from [1].

Another simpler way is to represent the skeleton sequence as an image. In this case, the spatial and temporal information are mixed. The sections below demonstrate the results of two experiments training with two types of images, i.e. 3-channel skeletal joints, and 1-channel skeleton sequence. The architecture of the CNN is shown in Figure 2. As the network is custom, it has to be trained from scratch.



```
ConvNet(
  (layer1): Sequential(
    (0): Conv2d(3, 16, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (layer2): Sequential(
    (0): Conv2d(16, 32, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (fc1): Linear(in_features=2048, out_features=500, bias=True)
  (fc2): Linear(in_features=500, out_features=4, bias=True)
)
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 16, 32, 32]           1,216
       BatchNorm2d-2           [-1, 16, 32, 32]              32
              ReLU-3           [-1, 16, 32, 32]               0
         MaxPool2d-4           [-1, 16, 16, 16]               0
            Conv2d-5           [-1, 32, 16, 16]          12,832
       BatchNorm2d-6           [-1, 32, 16, 16]              64
              ReLU-7           [-1, 32, 16, 16]               0
         MaxPool2d-8             [-1, 32, 8, 8]               0
            Linear-9                  [-1, 500]       1,024,500
           Linear-10                    [-1, 4]           2,004
================================================================
Total params: 1,040,648
Trainable params: 1,040,648
```

Figure 2: Summary of 2D CNN used for training.

1. Experiment 1: 3-channel images of skeletal joints as input

Since the skeletal joints are 2D in this project, the third channel is obtained by taking the difference of two frames in x direction. The interval of frame difference is 10 frames, which corresponds to one second and is less than a gait cycle. The reason of taking the frame difference is that many gait features like stride length can be inferred from the x direction. A sample image is shown in Figure 3, in which a column of the image array contains the joints of one frame. Note that the joints are arranged in a way similar to [2] (Figure 4(d)), such that the joints of each part of the body are closer to each other.

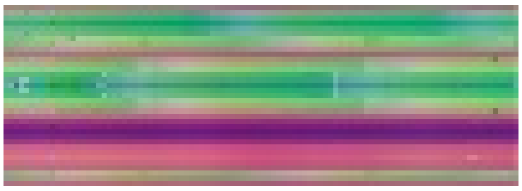


Figure 3: A sample of skeleton image.

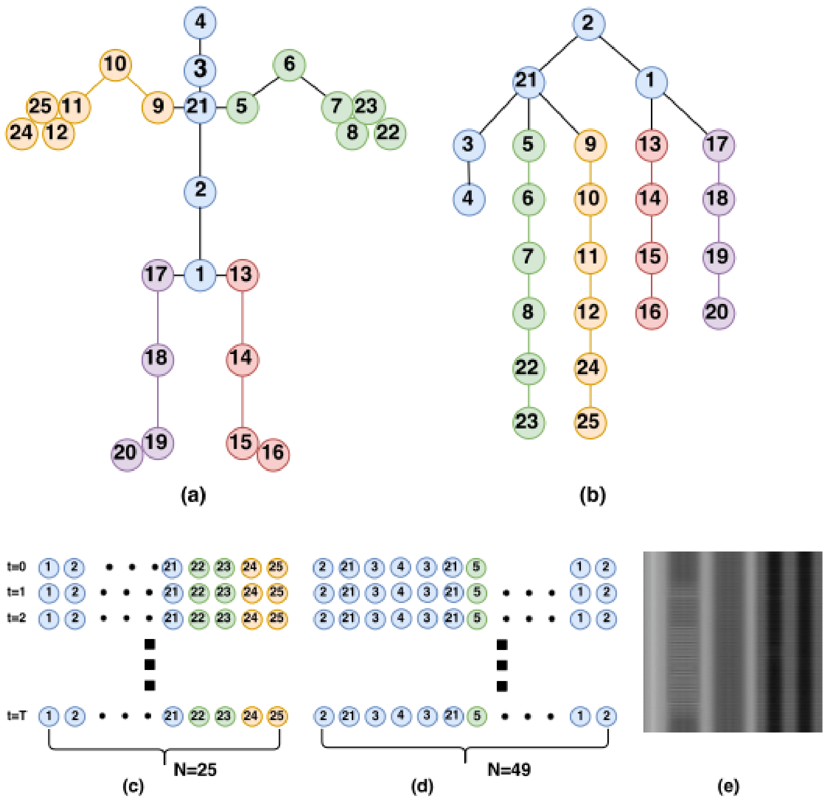


Figure 4: Tree Structure Skeleton Image (TSSI) proposed by [2].

- **Dataset:** Each data sample contains the coordinates of 18 skeleton joints from 100 video frames (around two gait cycles) and a locomotion score as the label. The dataset has 501 samples, divided into training and validation sets with a 70/30 ratio.
- **Keypoints:** Figure 5 shows the 18 keypoints (skeleton joints) from by pose estimation. Note that the coordinate of all the points are calculated from the point 5 instead of the absolute position in the frame.
- **Result:** The training and validation loss and accuracy curves of different image sizes are displayed in Figure 6 and 7. As can be seen in the figures, the training result just improves a little within the first 10 epochs. The image size seems to have no effect on the accuracy.

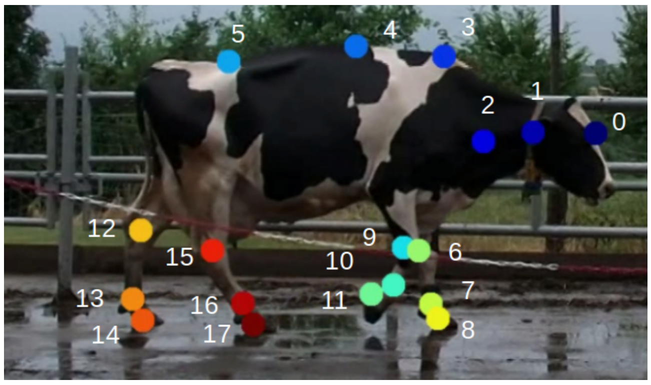


Figure 5: The 18 keypoints extracted from videos for lameness detection.

- **Notes:**
  - It seems to be an under-fitting problem, or the data are noisy and not informative enough.
  - Pre-trained networks with complex architecture have been applied as well, but the result is even worse.

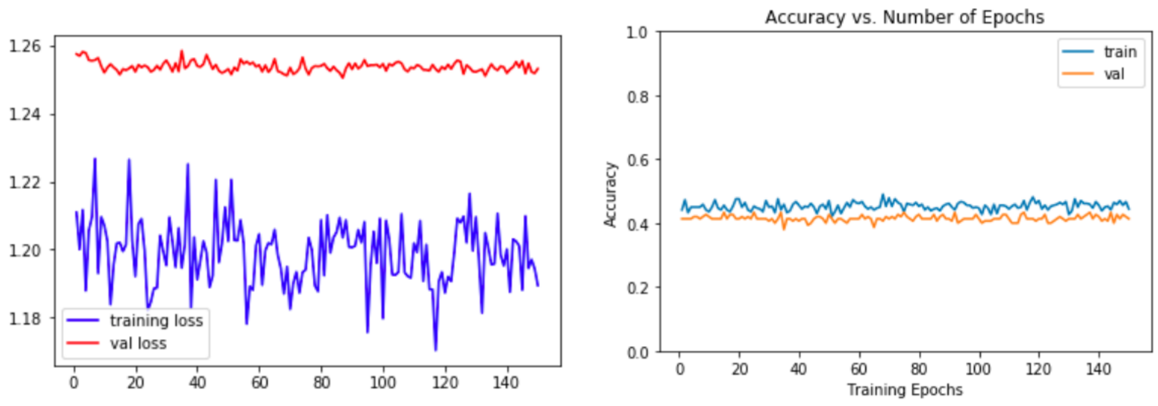


Figure 6: Training/validation loss (left) and accuracy (right) with input size of 32.

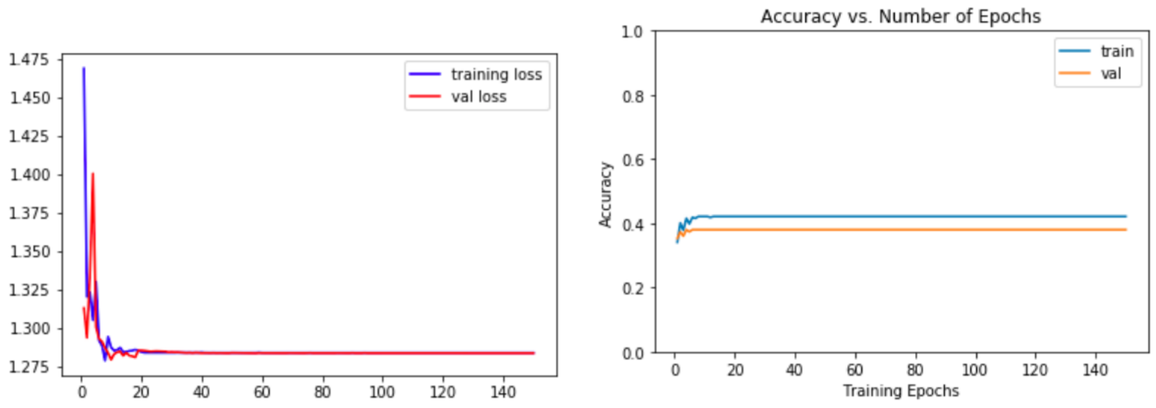


Figure 7: Training/validation loss (left) and accuracy (right) with input size of 224.

2. Experiment 2: Skeleton sequence as input

This experiment used the skeleton sequence as input data. Each input image has only one channel and contains 30 frames of joints. Figure 8 shows a sample image.

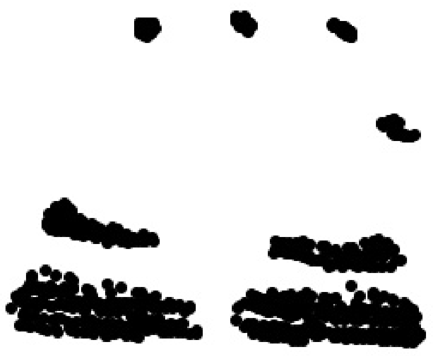


Figure 8: Skeleton sequence. The cow is facing rightward.

- **Dataset:** The dataset has 501 samples, divided into training and validation sets with a 70/30 ratio.
- **Keypoints:** The 18 keypoints shown in Figure 5 were considered.
- **Result:** Figure 9 shows the training/validation loss and accuracy curves. Just like previous experiment, the training result only improves a tiny amount in the first 10 epochs.

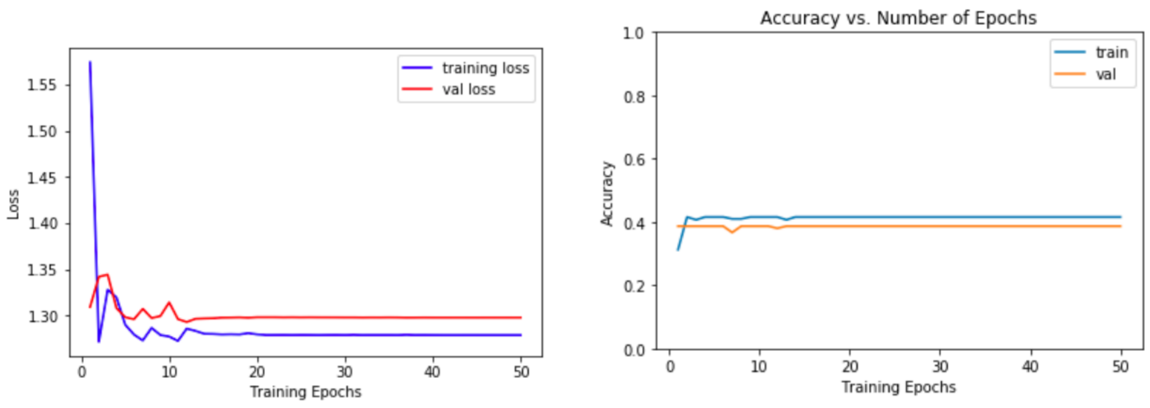


Figure 9: Training/validation loss (left) and accuracy (right) with input size of 224.

3. Discussion

It turns out that the 2D CNN cannot learn the features to classify the level lameness of cows, which seems to be the issue of under-fitting. Different pre-trained networks with complex architecture provided in Pytorch, such as Alexnet and VGG, have been tested by training the last layer. The training curves does not show much improvement. The noise containing in the data, which is an issue of pose estimation, can be the cause as well. Hence, the next steps include improving the pose estimation result and trying other methods which can extract more useful spatio-temporal information and are less prone to the noisy data.

Bibliography

[1] Y. Du, Y. Fu, and L. Wang, "Skeleton based action recognition with convolutional neural network," in *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*, pp. 579–583, IEEE, 2015.

[2] Z. Yang, Y. Li, J. Yang, and J. Luo, "Action recognition with visual attention on skeleton images," in *2018 24th International Conference on Pattern Recognition (ICPR)*, pp. 3309–3314, IEEE, 2018.

[3] H.-H. Pham, L. Khoudour, A. Crouzil, P. Zegers, and S. A. Velastin, "Learning to recognise 3d human action from a new skeleton-based representation using deep convolutional neural networks," *IET Computer Vision*, vol. 13, no. 3, pp. 319–328, 2019.