Protocol 21: RNN Training

31.05.2019

Recurrent Neural Networks (RNN)

1. Introduction

The hierarchical recurrent neural network ([1]) was used to train the skeleton data for lameness detection. As the training last week showed constant loss and accuracy throughout the training process, some strategies have been applied to find out the causes.

2. Experiment

- **Dataset:** The original dataset contains 501 samples, each of which contains the coordinates of 25 skeletal joints (Figure 1) from more than 100 video frames and a locomotion score as the label. As in the previous protocols, the locomotion scores fall into four classes. The dataset is augmented by mirroring and stretching, such that the amount of data is tripled. The streching is only applied to x coordinate based on the assumption that the cow is standing on level ground in the frames.

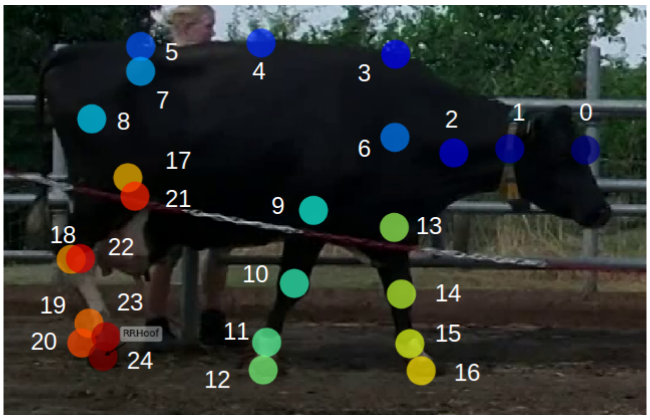

Figure 1: The 25 keypoints of cow skeleton.

- **Keypoints:** Figure 1 shows the 25 keypoints (skeletal 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. Besides, the values of x and y coordinates are divided by the frame width and height, respectively.
- **Strategies:**
  - Architecture: simple (double-layered) [2], complex (hierarchical architecture) networks
  - Different labels: original, fake labels (constant, random labels)
  - Different number of frame sequences (Figure 2): 30, 100 (In case a video has less than 100 frames, the frames from the start are appended to the end.)
  - Batch size: 4, 16, 64
- **Result:** Experiments with different parameters all lead to constant loss/accuracy after a few epochs. The network always predicts the same class.

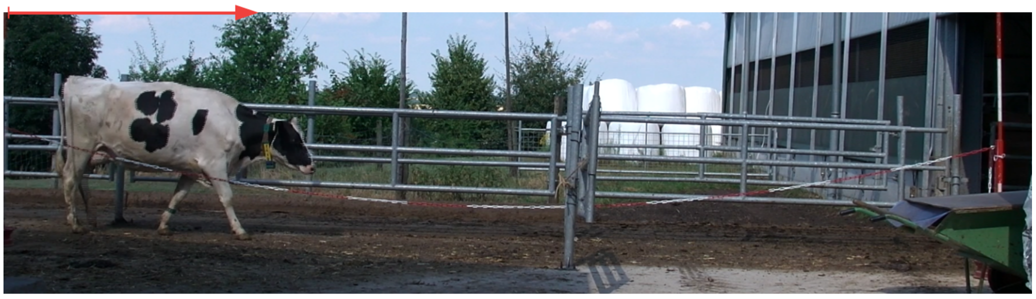

Figure 2: The scene of walking cow. The red arrow at the top left shows the distance that a cow moves in around 30 frames, corresponding to 1.5 seconds in which a cow can complete a gait cycle.

3. Discussion

The issue of stagnation of training still remains. The probable causes mentioned in the last protocol are:

- Bugs in network or training codes

  One concern was that there are some errors in the codes, such that the network could not be trained properly. A simple network from [2] was applied, but the issue still remains. This shows that there are some other causes, but does not eliminate the possibility of bugs in the codes, however.

- Data: poor data quality (noisy coordinates), unnormalized data between layers

  The noisy data arise mainly from pose estimation with occlusion, so is affected by choice of nunmber of frame sequence. 30 frames may have better data quality but with less information than 100 frames. Thus the pose estimation still needs to be improved. Data normalization between layers has not been checked yet.

- Data distibution: imbalanced data (prdominant class 1)

  One strategy was to use fake labels. If all the labels are the same (any integer in [0,3]), the network can predict all of them. If the labels are random (meaning there is no imbalanced data), the network still predicts the same class for all the data rather than random guess. This can implicitly indicate that either the network is not able to extract useful information from the data, or the data do not contain enough information for the network to make prediction.

- Unsuitable haperparameters and weight initialization

  Different values of learning rate, batch size, weight decay have been tried, but it seems these are not the main cause of learning stagnation.

- Inappropriate network architecture

  Since only two types of network were tested, it is still possible that the chosen network architecture is not appropriate.

Reference

[1] Y. Du, W. Wang, and L. Wang, Hierarchical recurrent neural network for skeleton based action recognition, in Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 11101118, 2015.
[2] https://github.com/yunjey/pytorch-tutorial/blob/master/tutorials/02-intermediate/bidirectional_recurrent_neural_network/main.py