**Improving gait classification in horses by using inertial measurement unit (IMU) generated data and machine learning**

* **Up sampling the 10 Hz Magnetometer Readings:** Alternatively, you could sample the magnetometer data from 10 Hz to 100 Hz to match the accelerometer and gyroscope data. This can be done using interpolation methods, such as linear interpolation, to estimate the magnetometer readings at the 100 Hz timestamps.
* **Get consistent sample rate from all sensors with timestamps.**
* **Normalization Gyro Data:** Normalize the data from each sensor to ensure that all input features have a similar scale. This can improve the convergence speed of the LSTM and its overall performance.
* **Set Maximum Ranges for Sensor readings:** For better accuracy and normalization.
* **Windowing:** Segment the continuous time series data into smaller fixed-size windows. Each window will serve as one input sequence for the LSTM. The size of the window (number of time steps) will depend on the temporal patterns you expect the LSTM to learn.

**Deep-Learning-Based Character Recognition from Handwriting Motion Data Captured Using IMU and Force Sensors**

* **Position of the IMU:** Must produce the maximum motion for a given movement, more features can be extracted that way.
* **Pause after repetition:** This brief pause was useful for trimming the raw data to small sizes of datasets before training.
* **Window Considerations:** LSTM networks can handle sequences of data, but the efficiency and effectiveness can diminish if the sequences are too long. By using a fixed time length (in this case, 1.3 seconds or 200 data samples), the model deals with a manageable size of data that represents one character. This helps in reducing complexity and improving the training efficiency of the network.

Data is fed into an LSTM in sequences. For handwriting recognition, a sequence could be a time-series representation of one character, including its position, pressure, angle, and other relevant features captured over time.

* **Threshold Acceleration Value:** To differentiate between inactivity and motion. Can help save computation power as the LSTM does not always keep running.
* **Data Restructuring:** Restructure the data into a virtual image to pass to a CNN.

**Additional Information:**

* **Labeling (Many to One and Many to Many):   
  Many to One:** Windowing will be necessary to pass segments into the LSTM. Would have to decide when to start or end that window based on features extraction. Features of the curve can be used to decide the start/end of the curl. Based on the start/end of the curl, the segment window will be extracted, passed through the LSTM and classification will be made using a Fully Connected Deep Neural Net at the end of the last sample in the data segment.

1. Lower Computational Load: Since the model only needs to generate a single output after processing the entire sequence, the computational load is typically lower than in a many-to-many setup.
2. Reduced Memory Usage: Memory usage is minimized as the device only needs to store a single final output, rather than an output for each time step.
3. Lower Power Consumption: The reduced computational load and memory usage translate to lower power consumption, which is ideal for battery-operated devices.
4. Potentially Lower Latency: Depending on the sequence length and model complexity, the many-to-one approach may offer lower latency since it processes the entire sequence before making a classification.

**Many to Many:** Labelling all the data collected at all timestamps. The output can be generated at each timestep.

1. Higher Computational Load: Generating outputs at each time step requires more computations, which can be taxing on devices with limited processing capabilities.
2. Increased Memory Usage: Storing intermediate states and outputs for each time step can quickly consume the limited memory available on embedded devices.
3. Power Consumption: The increased computational load and memory usage can lead to higher power consumption, which is a critical concern for battery-powered devices.
4. Latency: Processing time is crucial in real-time applications. The many-to-many approach might introduce latency that could be problematic for time-sensitive tasks.