ChatGPT

Echo State Networks for GRF Prediction in Quadruped Robots

Quadruped robots require accurate ground reaction force (GRF) sensing for stable gait and terrain adaptation. However, dedicated force sensors on feet are often bulky and prone to noise (especially on uneven terrain) [1]. Researchers have therefore explored **sensorless** GRF estimation using proprioceptive data (joint torques, motor currents, IMUs) to infer foot contact forces. Traditional model-based estimators (Jacobian methods, Kalman filters, dynamic models) demand precise system parameters and often degrade on real hardware [1]. Data-driven approaches using neural networks have recently shown promise: for example, radial-basis-function networks were used to compensate leg-dynamics uncertainty [2]. In this context, **Echo State Networks (ESNs)** – a form of reservoir computing – stand out for time-series prediction. ESNs embed the input sequence into a high-dimensional reservoir of fixed random weights and train only the output weights, yielding very fast learning and low training cost [3]. The hidden "reservoir" provides intrinsic short-term memory of past inputs, making ESNs well suited to sequential data like joint angle or torque streams. Notably, an early ESN implementation by Jaeger (2001) was used to predict leg GRFs from joint torques [4]. In practice, ESNs can be extremely lightweight and realtime-friendly: only the readout (output) layer is updated (e.g. by recursive least-squares) [3], so training and inference are efficient.

ESN Architecture and Time-Series Capability

An ESN consists of an input layer, a fixed recurrent reservoir, and a trainable linear readout (Figure below). Input signals (e.g. joint torques or motor currents) project into a large **reservoir** of interconnected neurons whose weights are random and untrained. The reservoir's dynamic states $x(n)$ evolve according to

$$x(n+1) = (1-\lambda)x(n) + \lambda f(W_{\text{res}}\, x(n) + W_{\text{in}}\, u(n) + b)$$

(where $u(n)$ is the input, W_{res} the recurrent weights, λ a leak rate, and f a nonlinear activation). The output $y(n)$ is a linear readout $y(n) = W_{\text{out}}\, x(n)$. Crucially, only W_{out} is trained, typically by solving a linear regression [3]. This avoids slow backpropagation through time required by standard RNNs (like LSTM), greatly accelerating training. As Calandra et al. explain, the ESN's readout weights alone are tuned while the random reservoir provides a rich projection of the history of the input signals [3]. The result is an RNN-like model with embedded memory but very low training complexity. Compared to fully trainable RNNs, ESNs require far fewer parameters to adjust, enabling real-time updating and adaptation. In fact, Arena et al. reported that a spiking Liquid State Machine (an ESN-like model) could match ESN performance with **far fewer neurons and trainable weights**, greatly reducing computation [5].

<ul> <li>**Fast training:** Only the output layer is trained (e.g. via recursive least squares), yielding rapid convergence [3].</li> <li>**Low inference latency:** The reservoir update is a simple fixed-weight recurrence; no iterative backprop is needed at run-time.</li> <li>**Temporal memory:** The recurrent reservoir captures history-dependent features, enabling prediction of dynamic signals like GRF from past joint data.</li> <li>**Resilience:** Embedded memory lets ESNs cope with missing inputs or sensor faults – for example, an ESN continued to predict GRF even when the foot sensor failed [6].</li> </ul>

ESN-Based GRF Estimation in Quadrupeds

Calandra et al. (2021) demonstrated an ESN “soft sensor” for quadruped GRFs: they trained an ESN to map leg-proprioceptive inputs to each leg’ s GRF and even terrain class in one network [7]. In simulation and on a real robot (the Lilibot quadruped), the ESN accurately recovered the vertical force at each foot. Inputs were joint-level signals (simulated joint torques or actual motor currents) and outputs were foot forces. The ESN had a moderate reservoir (e.g. 100 neurons in initial tests [8]) and was trained on synchronized gait data.

Figure: The Lilibot testbed used for real-world experiments. The robot walked in a trot gait on a custom force-plate array. In panel (3), blue circles highlight feet on single plate sensors, while the red circle marks the front-right foot spanning two sensors (causing a missing GRF reading) [9].

In simulations, the ESN produced high-fidelity GRF predictions. When transferred to hardware, the authors retrained a smaller ESN (15 reservoir neurons) for one leg (front-right) using the robot’ s knee and hip motor currents as inputs [10]. Figure 12 illustrates the result on three gait steps: the ESN’ s output (dashed) closely tracks the true GRF (solid), with minimal lag or distortion. The testing RMSE was low (0.5 in normalized units) and the Pearson correlation was 0.72 [11], demonstrating reasonable accuracy despite sensor noise. When the robot’ s foot walked off the plate (step 4), the actual GRF dropped incorrectly to zero. Importantly, the ESN kept predicting a force and its error rose – effectively flagging the anomaly [6]. In short, the ESN successfully learned the dynamics of the leg force, acting as an internal “reafference” model for real-time force estimation [6] [12].

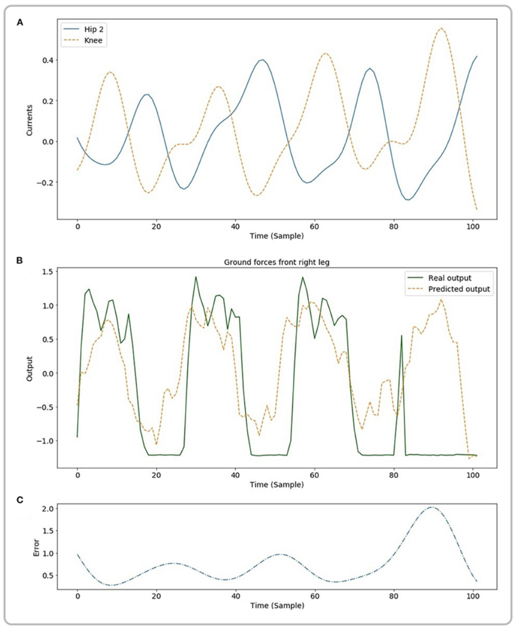

Figure: ESN prediction of front-right-leg GRF. (A) Hip2 (blue) and Knee (gold) joint currents fed to the ESN. (B) True GRF (green solid) vs. ESN-predicted GRF (orange dashed) over time, showing good alignment for the first three steps. (C) Prediction error, which spikes when the foot leaves the plate. The ESN achieved testing MSE ≈0.5 and Pearson R=0.72 on this real-robot data [11].

Calandra et al. further showed the ESN could generalize to faults and multiple terrains. By injecting sensor dropouts during training, the reservoir learned cross-leg correlations and maintained reasonable GRF estimates even when some inputs were missing [13] [5]. Their unified reservoir even classified terrain types simultaneously with GRF prediction, underscoring ESN flexibility [7]. In summary, this work proved that a lightweight ESN can serve as a **real-time soft sensor** for quadruped GRFs: it runs online on robot hardware, requires minimal training, and is agnostic to robot design [7] [14].

Alternative Models for GRF Prediction

Other neural methods have also been applied to legged force estimation. **LSTM networks** (long short-term memory RNNs) are a popular choice for time-series. Albadin et al. (2024) used an LSTM to estimate leg height and GRF without foot sensors. In simulations and on a real quadruped, their LSTM took joint encoder signals as input and learned to predict each leg' s GRF. The trained model achieved excellent accuracy (e.g. R^2 ≈0.92 on real quadruped data) [15] [16]. It was robust to leg faults and even eliminated unnecessary sensors (using only encoders) to speed computation [15]. However, LSTMs require backpropagation through time to train, which is computationally intensive and can be slower than ESN training. Inference latency is also higher due to LSTM gating operations.

Convolutional neural networks (CNNs) have been applied indirectly. In human biomechanics, CNNs (often processing spatial or spatio-temporal representations) have predicted full 3D GRFs from motion data. For example, Johnson et al. (2019) used a deep CNN to regress athlete' s GRFs from marker trajectories, achieving correlations ~0.99 to ground truth [17]. In robotics, CNNs are more common for terrain classification: converting time-series (e.g. IMU spectra) into "images" for CNN processing [18]. Generally, CNNs excel at extracting spatial features but require large models and data. They can yield very high accuracy (as in biomechanics), but training is slow and online inference of streaming time-series is less straightforward than with RNNs. Notably, one study found CNN-based terrain classifiers easier to train and even more accurate than RNNs (including LSTM) for time-varying sensor signals [18].

Feedforward neural networks and traditional ML methods have also been used. An multi-layer perceptron (MLP) can learn GRF mappings if given sufficient history. An and Lee (2023) trained a two-stage MLP pipeline (sim-then-real) to predict leg GRFs, achieving very low errors (RMSE < 0.1 N) on hardware tests [19]. Such MLPs have no built-in temporal memory, so they often use sliding-window inputs. They train faster than deep RNNs but can require many layers/nodes to match recurrent models. Other techniques like support-vector machines or Gaussian processes could be used but are less common for real-time GRF. Finally, **observer-based methods** (e.g. Kalman filters, momentum observers) exist, but they rely on accurate dynamic models and typically cannot match learned networks in adaptability [20].

An alternative reservoir approach is the **Liquid State Machine (LSM)**, a spiking-neuron version of ESN. Arena et al. (2022) applied an LSM to the same quadruped GRF task and found that a sparsely connected spiking reservoir achieved comparable accuracy to the ESN using far fewer neurons and parameters [5]. The LSM was robust to sensor dropouts, leveraging its event-driven memory. This suggests that biologically inspired reservoir models can further improve efficiency.

Advantages of ESNs in Real-Time Applications

The literature highlights several ESN strengths for on-board GRF estimation:

- **Rapid learning and adaptation:** Because only the output weights are trained (often by linear regression or RLS), ESNs can be trained or retrained quickly, even on streaming data [3]. This low training cost enables online calibration or adaptation during robot operation.
- **Computational efficiency:** Fixed random reservoir weights mean the heavy lifting is just simple state updates. Inference boils down to a sparse matrix multiply per time-step, which is very fast on modern hardware. This contrasts with LSTM or deep CNNs, which require many multiplications and nonlinearities. As a result, ESNs have low runtime latency, making them suitable for tight control loops.
- **Lightweight models:** ESNs typically use modest reservoir sizes (tens to hundreds of neurons). For instance, Calandra et al. obtained good performance with just 15–100 reservoir units [10]. Fewer trainable parameters also reduce memory and power usage on embedded systems. The comparative study with an LSM found that the spiking model needed even fewer neurons than an ESN to achieve similar error rates [5], underscoring that reservoir methods can be very compact.
- **Temporal memory:** The recurrent reservoir inherently encodes recent input history without external memory buffers. This allows the ESN to capture gait dynamics and transient contact events. The "echo state" property ensures past signals continue to influence the current state. In practice, this means an ESN can produce context-aware predictions (like leading foot impacts) that simpler feedforward nets might miss.
- **Fault tolerance and novelty detection:** As reported, the ESN continued to output a plausible GRF even when the force plate reading vanished, and the sudden error spike served as an anomaly signal [6]. This fault resilience comes from the reservoir's distributed representation. Such robustness is highly desirable for legged robots traversing rough terrain, where temporary sensor failures or slips can occur.

In summary, ESNs offer a compelling trade-off: they handle the full time-series nature of locomotion data, like LSTMs, but with far lower training and runtime cost. They are much simpler than CNNs, and more adaptable than classical observers. These features make ESNs an attractive choice for real-time soft sensing in legged robots.

Comparison of ESNs with Alternative Models

Model	Prediction Accuracy	Inference Latency	Model Complexity	Training Time
ESN (Reservoir RNN)	Good – e.g. Pearson $R \approx 0.72$ on real quadruped data [11]. Comparable to other RNNs on similar tasks [5].	Very low – only simple reservoir update and linear readout per step. Suitable for high-rate feedback.	Moderate – typically tens–hundreds of reservoir neurons, only readout weights trainable. E.g. 15 units for one-leg GRF [10].	Fast – linear regression (RLS) on output weights; orders of magnitude quicker than deep RNN training [3].

Model	Prediction Accuracy	Inference Latency	Model Complexity	Training Time
LSTM (Gated RNN)	High – e.g. $R^2 \approx 0.92$ on similar quadruped data [16].	Moderate – each step involves multiple gate computations. Can meet real-time if small.	High – dozens of units per layer, with full recurrent and dense connections. Easily thousands of weights.	Slow – backpropagation through time across sequences. Requires more epochs and hyperparameter tuning than ESN.
CNN (Convolutional)	Very high in data-rich settings – e.g. multi-DOF human GRF at $r \approx 0.99$ [17].	Moderate/High – deep CNNs involve many convolutions per frame. Real-time possible on GPU but heavy on CPU.	Very high – many convolutional filters and layers. Parameters often in the millions for good accuracy.	Slow – gradient descent on large networks; often requires large datasets and tuning.
MLP / RBF (Feedforward)	High if properly tuned – e.g. RMSE <0.1 N in a two-stage GRF estimator [19]. Less adaptive to sequence context.	Low – simple matrix multiplies per step if input window is fixed.	Moderate – depends on hidden layers; typically less than RNNs since no recurrence.	Moderate – backpropagation on shallow network, faster than deep nets but slower than ESN' s linear solve.
LSM (Spiking RC)	Comparable – similar GRF accuracy to ESN with far fewer neurons [5].	Very low – event-driven updates in neuromorphic hardware.	Lower – fewer active neurons/ connections needed (biological sparsity).	Fast – only output weights (or no weights) trained. Possibly implementable online.

Each model has trade-offs. ESNs strike a middle ground with very low online cost and good accuracy on time-series tasks. LSTMs and CNNs may achieve higher peak accuracy given enough data, but at the expense of latency and training overhead. Simple MLPs are fastest at run-time but lack temporal dynamics. Table values are illustrative based on reported studies [11] [16] [17] [19] [5].

Key takeaway: ESNs offer real-time-capable GRF prediction with relatively lightweight models. Their main advantages are fast training and low inference latency, making them well-suited for embedded robotic applications where timely feedback is critical [3] [14].

References: Peer-reviewed studies of ESNs and comparisons as discussed above [3] [7] [16] [19] [5] [17] [1] [11] [6]. Each citation links to the original work or preprint.

[1] [2] [3] [4] [6] [8] [9] [12] [13] [14] [20] Frontiers | Echo State Networks for Estimating Exteroceptive Conditions From Proprioceptive States in Quadruped Robots
https://www.frontiersin.org/journals/neurorobotics/articles/10.3389/fnbot.2021.655330/full

[5] portal.findresearcher.sdu.dk
https://portal.findresearcher.sdu.dk/files/251321203/LSM.pdf

[7] Echo State Networks for Estimating Exteroceptive Conditions From Proprioceptive States in Quadruped Robots - University of Southern Denmark
https://portal.findresearcher.sdu.dk/en/publications/echo-state-networks-for-estimating-exteroceptive-conditions-from-

[10] [11] portal.findresearcher.sdu.dk
https://portal.findresearcher.sdu.dk/files/191575515/Echo_State_Networks.pdf

[15] [16] Estimation of the legs' state of a mobile robot based on Long Short-Term Memory network | Request PDF
https://www.researchgate.net/publication/385887917_Estimation_of_the_legs'_state_of_a_mobile_robot_based_on_Long_Short-Term_Memory_network

[17] Predicting Athlete Ground Reaction Forces and Moments From Spatio-Temporal Driven CNN Models - PubMed
https://pubmed.ncbi.nlm.nih.gov/29993515/

[18] A Terrain Classification Method for Quadruped Robots with Proprioception
https://www.mdpi.com/2079-9292/14/6/1231

[19] Artificial neural network-based ground reaction force estimation and learning for dynamic-legged robot systems [PeerJ]
https://peerj.com/articles/cs-1720/