A Physics-Augmented Deep Learning Framework for Classifying Single Molecule Force Spectroscopy Data

Cailong Hua [1] **Sivaraman Rajaganapathy** [2] **Rebecca A. Slick** [3] **Joseph Vavra** [3] **Joseph M. Muretta** [3] **James M. Ervasti** [3] **Murti V. Salapaka** [1]

Abstract

Deciphering protein folding and unfolding pathways under tension is essential for deepening our understanding of fundamental biological mechanisms. Such insights hold the promise of developing treatments for a range of debilitating and fatal conditions, including muscular disorders like Duchenne Muscular Dystrophy and neurodegenerative diseases such as Parkinson's disease. Single molecule force spectroscopy (SMFS) is a powerful technique for investigating forces involved in protein domains folding and unfolding. However, SMFS trials often involve multiple protein molecules, necessitating filtering to isolate measurements from single-molecule trials. Currently, manual visual inspection is the primary method for classifying single-molecule data; a process that is both time-consuming and requires significant expertise. Here, we both apply state-of-the-art machine learning models and present a novel deep learning model tailored to SMFS data. The proposed model employs a dual-branch fusion strategy; one branch integrates the physics of protein molecules, and the other operates independently of physical constraints. This model automates the isolation of single-molecule measurements, significantly enhancing data processing efficiency. To train and validate our approach, we developed a physics-based Monte Carlo engine to simulate force spectroscopy datasets, including trials involving single molecules, multiple molecules, and no molecules. Our model achieves state-of-the-art performance, outperforming five baseline methods on both simulated and experimental datasets. It attains nearly 100% accuracy across all simulated datasets and an average accuracy of $79.6 \pm 5.2\%$ on experimental datasets, using only $\sim$30 training samples, surpassing baseline methods by 11.4%. Notably, even without expert annotations on experimental data, the model achieves an average accuracy of $72.0 \pm 5.9\%$ when pre-trained on corresponding simulated datasets. With our deep learning approach, the time required to extract meaningful statistics from single-molecule SMFS trials is reduced from a day to under an hour. This work results in SMFS experimental datasets from four important protein molecules crucial to many biological pathways. To support further research, we have made our datasets publicly available and provided a Python-based toolbox (https://github.com/SalapakaLab-SIMBioSys/SMFS-Identification).

1. Introduction

Many biological processes depend on controlling mechanical forces achieved via the folding and unfolding of domains in molecules like titin (Rief et al., 1997; 1998; Oberhauser et al., 2001), dystrophin and its homologue utrophin (Rajaganapathy et al., 2019; Ramirez et al., 2023), neurotoxic proteins (Hervás et al., 2012), and extracellular matrix protein tenascin (Oberhauser et al., 1998). For example, dystrophin and utrophin work as molecular shock absorbers that limit myofiber membrane damage when undergoing reversible unfolding upon muscle stretching and contraction (Ervasti, 2007). Evidently, studying mechanical properties of single molecules can provide vital insights into mechanisms of debilitating diseases. Instruments such as optical tweezers (Ashkin et al., 1986) and atomic force microscopes (AFMs) (Binnig et al., 1986) have enabled single molecule force spectroscopy (SMFS), where molecular forces in the femto

[1]Department of Electrical and Computer Engineering, University of Minnesota - Twin Cities, Minneapolis, MN 55455 [2]Department of Artificial Intelligence and Informatics, Mayo Clinic, Rochester, MN 55905 [3]Department of Biochemistry, Molecular Biology and Biophysics, University of Minnesota - Twin Cities, Minneapolis, MN 55455. Correspondence to: Cailong Hua <hua00023@umn.edu>, Murti V. Salapaka <murtis@umn.edu>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

to nano-Newton range, over sub-nanometer to micrometer distances, can be measured and studied. In an SMFS experiment, measurements from a force probe are recorded while it is made to interact with molecules of interest. Since the size of a typical force probe is orders greater than a typical bio-molecule, the probe may come in contact with one or more molecules. Measurements obtained from interactions involving more than one molecule confound the interpretation of results and are a significant challenge in characterizing the behavior of single molecules. How to identify and isolate measurements and data that result from a single-molecule is an important objective for SMFS.

The use of chemical functionalization of probes, along with molecular fingerprints has emerged as an approach for identifying single molecule trials (Yang et al., 2020a). Chemical functionalization modifies the surface of the force probe and substrate to enable site-specific attachment of molecules, where fingerprints are well-characterized molecules that yield distinct unfolding patterns. Here fingerprints in the trials can be leveraged to discern single molecule trials from trials that result from multiple molecules. However, surface chemical functionalization is time-consuming, often taking at least 6 hours (Zimmermann et al., 2010), and demands careful handling and practice. Moreover, advanced filtering techniques, informed by an understanding of all molecules involved in the complex, are essential for effectively identifying single molecule trials (Yang et al., 2020a).

In contrast, conducting experiments without functionalizing probes and introducing fingerprints into the native molecule has significant advantages. Probes without functionalization are easier and less expensive to manufacture, and biomolecules without fingerprints engineered into their structure are easier to synthesize. Moreover, there is greater confidence that the experimental data characterizes the unaltered native bio-molecule without any confounding effects introduced by fingerprints. Despite these advantages, when there are no fingerprints, distinguishing data that result from single molecules and multiple molecules is more challenging. Currently, prevalent accepted method for distinguishing the data is based on visual inspection, which is a time-consuming process that demands a high level of expertise (Bornschlögl & Rief, 2011; Lyubchenko, 2018). Additionally, trials need to be collected from a large number of experiments, not only to ensure statistical confidence but also because the molecule concentration is generally lowered to minimize the possibility of involving multiple molecules (Ramirez et al., 2023; Oberhauser et al., 2001). These factors hinder obtaining precise statistics of single molecular trials without functionalization and the generation of a large, annotated dataset suitable for training deep learning models.

A typical traditional workflow for SMFS experiments involves purifying protein molecules from expression systems (either bacterial or insect), preparing samples with the target protein molecules, and setting up the AFM to automatically perform multiple pulls to obtain numerous SMFS trials. Subsequent to obtaining trial data, an expert manually filters which curves correspond to single molecules. Each session typically involves approximately 2,000–5,000 trials, requiring the expert to meticulously sift through the data to filter out curves that are not from single molecules. This process can take between 12 and 24 hours. To ensure reliable assessment, the molecule is expected to be expressed at least thrice, with each expression resulting in multiple sessions, making the cumulative time investment significant. This article aims to reduce this time to less than one hour.

In this work, we introduce state-of-art machine learning models and present a novel deep learning model that augments the unfolding physics of protein molecules to accurately classify SMFS data into three classes: 1) no molecule, 2) single molecule, and 3) multiple molecules. The model employs a dual-branch fusion strategy, one branch incorporating the physics of protein molecules and the other operating independently of physical constraints. To train and validate our approach, we present the first publicly accessible datasets, both simulated and experimental, obtained from non-specific pulling of multi-domain molecules: titin, utrophin, and dystrophin (Hua et al., 2024). The simulated datasets, comprising of SMFS trials that originate from a single molecule, multiple molecules, or no molecules, are created with a novel physics-based Monte Carlo engine. Extensive evaluations on these datasets, in comparison with five state-of-the-art baseline models, demonstrate efficacy of our proposed model. Specifically, our model achieves nearly 100% accuracy on the simulated datasets, outperforming baselines by 6%. When testing on experimental datasets, our model achieves average accuracies of $79.6 \pm 5.2\%$, surpassing baselines by 11.4%, with only approximately 30 training samples. Even without expert annotations on experimental data, our model can still achieve average accuracies of $72.0 \pm 5.9\%$ when pre-trained with corresponding simulated datasets. With this deep learning approach, the time required to extract meaningful statistics from single-molecule SMFS trials is reduced from a day to under an hour. It is expected that this pilot effort will stimulate significant activity with associated impact of ML methods on the understanding of protein folding and unfolding that emphasize mechanical properties.

2. Related work

Single molecule classification A 1D convolutional neural network trained using a triplet loss function (Hoffer & Ailon, 2015) was utilized to classify force curves into single, multiple, or no molecule classes, with reported accuracy

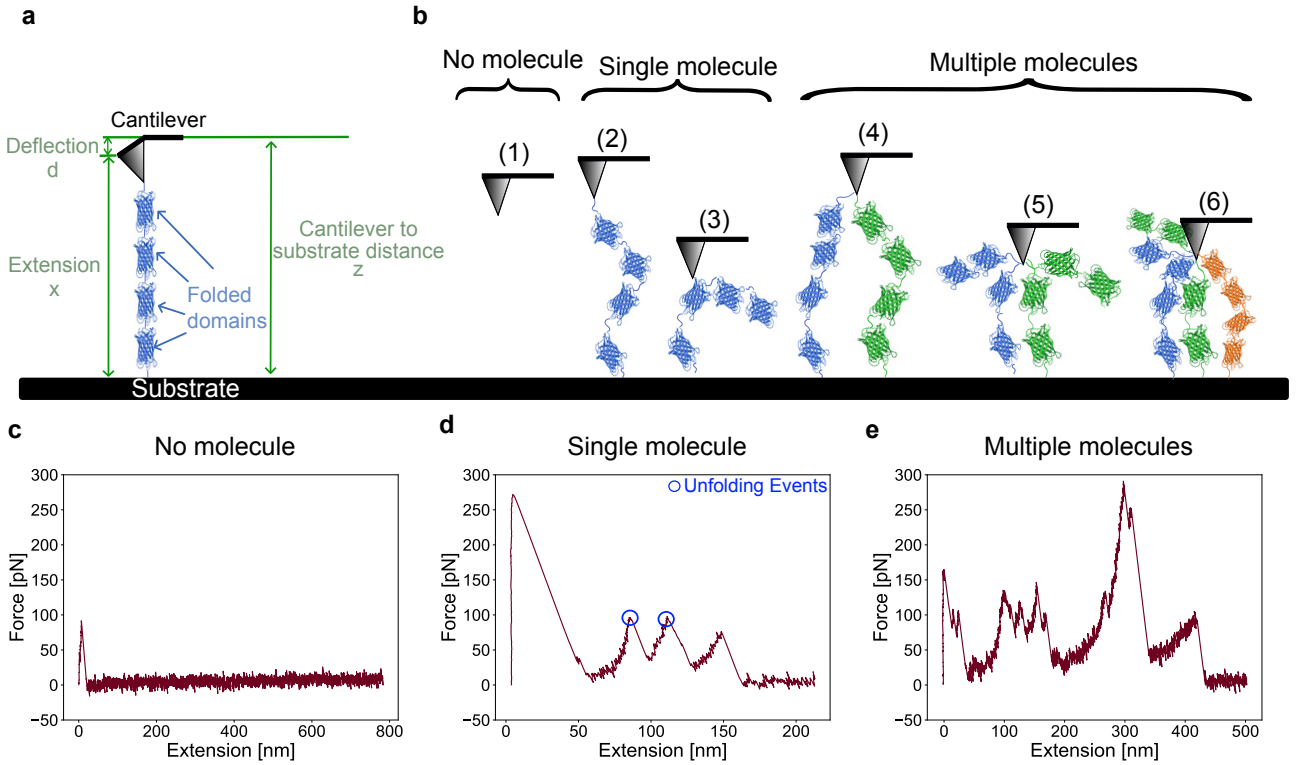

Figure 1. Illustration of AFM based SMFS. (a) Schematic showing the desirable case of a single protein molecule with four folded domains under tension between the tip of the AFM cantilever and the substrate. The deflection d and the separation between the cantilever and the substrate z are measured. The tensile force on the protein molecule is computed from the deflection d. (b) Depictions of possible scenarios, categorized into three classes: (1) no molecule present between the tip and substrate, (2-3) a single molecule or a section of a single molecule present between the tip and the substrate, and (4-6) multiple molecules or sections of multiple molecules between the tip and the substrate. (c-e) Show example force curves representative of the three different classes, with blue circles highlighting unfolding events.

ranging from $65 - 70\%$ (Waite et al., 2023). Moreover, a machine learning workflow was proposed to iteratively classify different unfolding pathways of single molecule curves (Doffini et al., 2023). However, these two datasets were collected with chemically functionalized probes and fingerprints. The chemical functionalization process is dependent on the specific molecule and thus cannot be made agnostic to the molecule under investigation. Moreover, in the first dataset, each single molecule curve contains only one unfolding event (Waite et al., 2023), simplifying the classification problem; the second dataset comprises images rather than time series data (Doffini et al., 2023), which introduces redundancy given that force curves are inherently time series data. There are currently no time series datasets available, which are from non-specific pulling of multi-domain protein molecules; most naturally occurring protein molecules have multiple domains. Here, we construct such datasets for classification purposes.

Time series classification (TSC) More than hundreds of time series classification (TSC) algorithms, including both non-deep learning methods (Bagnall et al., 2017) and deep learning methods (Fawaz et al., 2019; Wang et al., 2017), are present in prior-art. Although more than 80 different datasets from the University of California, Riverside (UCR) time series classification repositories (Dau et al., 2019) are evaluated with these methods, none of these datasets include SMFS data. Methods that do not use deep learning become computationally intensive and impractical to execute on large-scale datasets (Bagnall et al., 2017; Fawaz et al., 2019). In this article, we focus on deep learning methods to classify our SMFS datasets.

3. Problem formulation

We first describe an atomic force microscope (AFM) based SMFS setup. Here, a microcantilever with a sharp tip is pressed against a substrate on which the protein molecules under study are deposited. Under applied force, parts of one or more protein molecules are non-specifically attached

to the cantilever tip; characterized by a stochastic adhesion event (Leite et al., 2012). Upon retraction of the cantilever from the surface, sections of the protein molecules between the tip of the cantilever and the substrate experience a tensile force. The record of the force F experienced by the cantilever (and therefore the molecule) versus molecule extension x is known as a force curve, as depicted in Figure 1c-e. If only one protein molecule is present between the cantilever tip and substrate, the force curve unveils important mechanical properties of the protein molecule. We illustrate such a scenario in Figure 1a, where a single protein molecule with four folded domains is attached between the substrate and the tip of the cantilever. As the cantilever retracts, the molecule experiences mechanical tension, eventually causing a folded domain to unfold. The applied force drops abruptly when a domain unfolds, as highlighted by the blue circles in Figure 1d. This process continues until either all domains are unfolded or the connection between the cantilever and the substrate is broken (Rief et al., 1997), producing a saw-tooth pattern of force curves (Figure 1d).

In practice, the force curves can be categorized into one of three classes - 1) **No molecule**: where no molecule is present between the tip and substrate, 2) **Single molecule**: when only a single molecule or a section of a single molecule is present, or 3) **Multiple molecules**: where multiple molecules or sections of multiple molecules are present between the tip and substrate. Example experimental force curves corresponding to the three classes are depicted in Figure 1c, 1d, and 1e respectively. The force curves originating from multiple molecules typically exhibit larger unfolding forces than those with a single molecule (Figure 1e) and have a mixture of unfolding events that cannot be traced back to a specific protein molecule (Fig. 1b (4-6)), confounding useful interpretation. Here, excluding force curves with no molecule and multiple molecules is necessary to obtain accurate and interpretable data from SMFS. The identification of the *single molecule* force curves is challenging due to a number of reasons: 1) a large number of force curves (2000-5000) need to be collected in a single experiment since protein molecule capture success rates are kept at 1-5% (Oberhauser et al., 2001; Ramirez et al., 2023), 2) the study of a specific protein molecule involves at least three replications for confidence on results, 3) the force curves are corrupted from instrument measurement noise and intrinsic thermal noise of the molecules, and 4) often the protein molecules under investigation have no prior characterization, which makes the adjudication time consuming and difficult even for domain experts.

The i-th force curve, of length $T^{(i)}$, consists of force data $\mathcal{F}^{(i)} = [F_1^{(i)}, F_2^{(i)}, \dots F_{T^{(i)}}^{(i)}]$ and extension data $\mathcal{X}^{(i)} = [X_1^{(i)}, X_2^{(i)}, \dots X_{T^{(i)}}^{(i)}]$. Each force curve $(\mathcal{F}^{(i)}, \mathcal{X}^{(i)})$ is associated with a class label $Y_i \in \{0, 1, 2\}$, corresponding to one of the three classes. Given a dataset of n samples, $\mathcal{D} = [(\mathcal{F}^{(1)}, \mathcal{X}^{(1)}, Y^{(1)}), \dots, (\mathcal{F}^{(n)}, \mathcal{X}^{(n)}, Y^{(n)})]$, the goal is to design an effective deep learning model capable of accurately predicting the class label of a force curve.

4. Proposed model

We introduce **P**olymer **E**lastic **M**odels **N**eural **N**etworks (PemNN), a novel deep learning model designed to classify force curves as originating from no molecule, single molecule or multiple molecules. PemNN contains two branches, the force trace branch, which uses the force data $\mathcal{F}^{(i)}$, and the physics-based branch, which incorporates polymer elastic models (Section 4.1) with both extension and force data. Features extracted from these branches are fused using either early fusion (depicted in Figure 2) or late fusion strategies (see Section 4.2).

Both branches pass through a convolutional block, comprising a 1-Dimensional convolutional layer, batch normalization layer (Ioffe & Szegedy, 2015) and a Rectified Linear Unit (ReLU) (Nair & Hinton, 2010) activation layer. Convolutional layers have demonstrated compelling performance and efficiency in time series classification (Wang et al., 2017; Fawaz et al., 2019; Karim et al., 2019; Pham et al., 2022; Zhang et al., 2020; Zheng et al., 2016; Foumani et al., 2021). Then features from the first convolutional block of both branches are fused with one of four methods detailed in Section 4.2. The fused output undergoes two additional convolutional blocks, and a Global Average Pooling (GAP) layer is applied to reduce parameters by averaging across the time dimension (Wang et al., 2017; Fawaz et al., 2019).

To further enhance temporal encoding, a long short-term memory (LSTM) (Hochreiter & Schmidhuber, 1997; Sundermeyer et al., 2012) layer is applied to each branch, followed by a dropout layer (Srivastava et al., 2014) to mitigate overfitting (Karim et al., 2019). Earlier studies have shown that augmenting convolutional layers with LSTM significantly improves performance in time series classification with only a modest increase in the number of parameters (Karim et al., 2019; Zhang et al., 2020; Hewamalage et al., 2021).

The outputs of the GAP and LSTM layers are concatenated and fed into a fully connected layer with three neurons, corresponding to the three classes: 1) no molecule, 2) single molecule, and 3) multiple molecules. A softmax activation function is used in the final fully connected layer, and the model is trained with categorical cross-entropy loss:

$$L(\mathcal{F}^{(i)}, \mathcal{X}^{(i)}) = -\sum_{j=1}^{3} Y_j^{(i)} log(\hat{Y}_j^{(i)}), \tag{1}$$

where $L(\mathcal{F}^{(i)}, \mathcal{X}^{(i)})$ represents the loss for classifying the force curve $(\mathcal{F}^{(i)}, \mathcal{X}^{(i)})$. Here, $Y_j^{(i)}$ is the label for class j

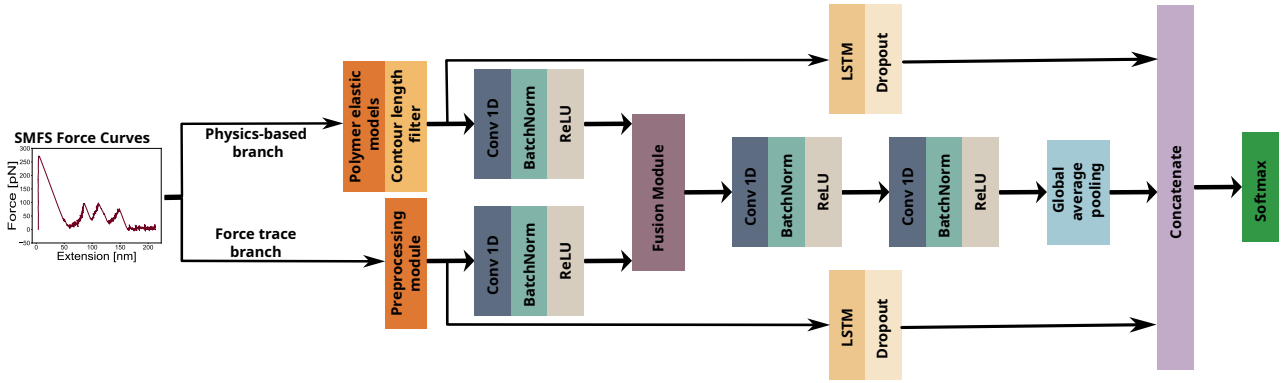

Figure 2. The overall architecture of the Polymer Elastic Models Neural Networks (PemNN) comprises a physics-based branch and a force trace branch, illustrated with the early fusion approach (the late fusion approach is shown in Figure 8 in the Appendix).

of i-th force curve ($Y_j^{(i)} \in \{0, 1\}$), and $\hat{Y}_j^{(i)}$ is predicted probability for class j of i-th force by the neural network.

4.1. Polymer elastic models

Polymers (proteins included) exhibit entropic elasticity that is well described by the worm-like chain (WLC) model (Bustamante et al., 1994) given by:

$$F = \frac{k_B T}{L_p} \left[\frac{1}{4\left(1 - \frac{x}{L_c}\right)^2} - \frac{1}{4} + \frac{x}{L_c} \right], \qquad (2)$$

where, F and x represent the force and extension respectively, k_B is the Boltzmann constant, T is temperature, L_c is the contour length, and L_p is the persistence length. The contour length L_c represents the maximal length of physically possible extension and the persistence length L_p quantifies the bending stiffness of the polymer. Other widely used polymer elastic models are described in Appendix A.1.

In the physics-based branch, the contour length $L_{cp}^{(i)}$ is estimated for each force-extension pair $\left(F_p^{(i)}, \mathcal{X}_p^{(i)}\right)$ using the WLC model with a fixed persistence length L_p. A subsequent filtering step selects P samples with $L_{cp}^{(i)} \in [0, M]$, where M is the filter threshold. If the number of qualified data points is less than P, sampling is performed with replacement.

4.2. Fusion module

The early fusion approach contains four methods: (i) Early-Sum: sums convolutional feature maps; (ii) Early-Max: selects the maximum value at each element from convolutional feature maps, (iii) Early-Wavg: computes a weighted sum of convolutional feature maps with learnable weights; and (iv) Early-Conv: concatenates convolutional feature maps along the filter dimension, followed by a convolution with a kernel size of 1.

With late fusion method (depicted in Figure 8 in the Appendix), features from the first convolutional block are independently processed through two additional convolutional blocks for each channel, with shared parameters. GAP layers are applied to the outputs of both branches, and their results are fused using one of the following methods (see Appendix B): (i) Late-Concat: directly concatenates the outputs of GAP layers; and (ii) Late-Gating (Liu et al., 2021): Concatenates GAP layer outputs with learnable weights.

5. Datasets

Datasets from four different protein molecules are employed. Out of these four protein molecules, only Titin I27O (Athena Enzyme Systems™) is an engineered protein molecule composed of eight repeats of the Ig 27 domain that serves as a reference protein for calibrating and validating our methods. The other three protein molecules come from natural proteins with considerable variations in their sequence and structure, dystrophin and utrophin. Dystrophin is a protein molecule expressed primarily at the muscle cell membrane, or sarcolemma, in striated muscle tissue. Deficiencies of this protein molecule lead to severe muscle wasting disorder like Duchenne muscular dystrophy (DMD), a fatal disease occurring in 1 out of 4000 male births (Mendell et al., 2012). Structurally, dystrophin is composed of four major domains: an amino terminal (NT) actin-binding domain (ABD1), a large central rod domain with 24 triple helical spectrin-like repeats (SLRs) interspersed with 4 hinge domains, including a second actin-binding domain (ABD2), a cysteine-rich domain and a carboxy-terminal (CT) domain (Figure 3a). Utrophin (Figure 3b) is a fetal homologue of dystrophin and is under active investigation as a dystrophin replace-

Table 1. Details of experimental datasets

DATASET	NUMBER OF CURVES PER CLASS	LENGTHS	DIFFERENT DAYS	PULLING SPEEDS $[nm/s]$
DDRS (WAITE ET AL., 2023)	[102,102,136]	400	NA	2000
TITIN I27O	[181,164,191]	736-4859	5	500, 1000, 2000
BACT UTRN-R3 (HUA ET AL., 2024)	[181,181,178]	957-9974	10	500, 1000, 2000, 5000
INSECT UTRN-R3 (HUA ET AL., 2024)	[175,166,200]	1777-8890	11	500, 1000, 2000
DYSN-R3 (HUA ET AL., 2024)	[191,185,177]	1659-4814	6	500, 1000, 2000

ment therapy for DMD. We include dystrophin and utrophin fragments encoding the NT through SLR 3 domains, referred to as DysN-R3 (Fig. 3c) and UtrN-R3 (Fig. 3d), respectively. Previous studies have demonstrated that the mechanical properties of UtrN-R3 are influenced by the expression system used, such as insect or bacterial cells (Ramirez et al., 2023; Hua et al., 2024). Consequently, we further categorize UtrN-R3 into insect UtrN-R3 and bact UtrN-R3 to reflect these variations. In summary, our four real protein molecules are: Titin I27O, bact UtrN-R3, insect UtrN-R3, DysN-R3. Additionally, we include the Discoidin Domain Receptors (DDRs) dataset (Appendix D.1) (Waite et al., 2023) to compare the performance of different deep learning methods.

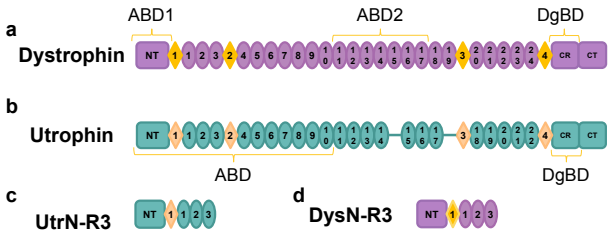

Figure 3. Diagrams of dystrophin and utrophin. (a) Full-length dystrophin. (b) Full-length utrophin. (c) UtrN-R3 construct (d) DysN-R3 construct. The ovals are spectrin-like repeats (SLRs); the diamonds are hinge domains; NT is the N terminus; CT is the C terminus; CR is a cysteine-rich domain; ABD1 & 2 are actin-binding domains; DgBD is the dystroglycan binding domain. Figure courtesy of previous study (Ramirez et al., 2023)

For each protein molecule, we have two datasets, one is simulation dataset generated with our physics-based Monte Carlo simulation engine described in Appendix A, and one is experimental dataset that is obtained from physical experiments conducted via AFM on real protein molecule samples. The simulation dataset, comprising 600 force curves, is generated for three classes; Class 0 has no protein molecule between the substrate and cantilever tip, Class 1 has one protein molecule, and Class 2 has two protein molecules attached. Different protein molecules are distinguished by different model parameters listed in Table 2. For the experimental data, as previously described in (Rajaganapathy et al., 2019; Ramirez et al., 2023; Hua et al., 2024), force curves were collected on different days and at various pulling speeds, as outlined in Table 1. As a result, the lengths of the force curves are different, and substantial variations exist even within the same class. Annotation is performed through visual inspection of the unfolding force curves.

6. Results and Discussion

The performance of our proposed model, PemNN, is evaluated in this section. We outline the evaluation setup in Section 6.1, followed by a comparison of PemNN's classification performance against baseline models in Section 6.2. Next, we analyze the functionality of the force trace branch and physics-based branch in Section 6.3. Finally, we assess performance under limited training data, a scenario frequently encountered in SMFS applications, in Section 6.4, and apply it to AFM data analysis through a comparison to non-machine learning methods in Section 6.5.

6.1. Evaluation setup

We evaluate our proposed model, PemNN, against five baselines: 1) Triplet network (Triplet) (Waite et al., 2023): The model designed for analyzing SMFS specific pulling data. 2) Fully convolutional neural networks (FCN) and 3) the residual networks (ResNet): Two of the highest performing deep neural networks on the UCR time series classification archive (Fawaz et al., 2019). 4) InceptionTime: The current state-of-the-art deep learning model on the UCR archive (Ismail Fawaz et al., 2020). 5) LSTMFCN: It outperforms FCN and ResNet on the UCR time series classification archive and demonstrates robust performance on multivariate time series classification (Karim et al., 2018; 2019). Further details about these baselines are provided in Appendix C.1.

The baselines were implemented using default parameters from sktime (Löning et al., 2019), which is a python framework for ML with time series data. For PemNN, the default configuration incorporates the WLC model, LSTM layers in both branches, and Early-Conv fusion method. An ablation study of PemNN's architecture is presented in Section

D.2, and additional implementation details are provided in Appendix C.2.

For each dataset, 20% of the data was used for training, and the remaining 80% was reserved for testing, as adjudication is a time-consuming task that requires significant expertise. Stratified resampling was applied to maintain class distributions. Each train-test split was seeded for reproducibility. Identical resamples were applied to all models within a single run, and performance was evaluated using overall accuracy of the test data across five runs.

6.2. PemNN classification performance

Per each dataset, we evaluated all models on testing data and ranked them based on their mean classification accuracy over five runs, assigning a rank of 1 to the most accurate model and 6 to the least accurate. The average ranking is then computed across all datasets, including both simulated and experimental testing sets for all protein molecules. These average rankings are summarized in the critical difference diagram (Demšar, 2006), as presented in Figure 4. PemNN achieves the lowest ranking of 1.4167, indicating that our model is more accurate than baseline models.

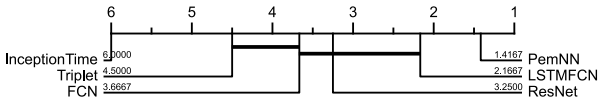

Figure 4. Critical difference diagram of different deep learning models across the simulated and experimental testing sets of all protein molecules based on average accuracies. The most accurate model is assigned a rank of 1, with a thick horizontal line representing a group of classifiers that do not exhibit statistically significant differences in accuracy.

For statistical analysis, we employed the Wilcoxon signed-rank test with Holm correction as the post-hoc test following the Friedman test (Fawaz et al., 2019; Demšar, 2006). In Figure 4, thick horizontal lines represent groups of models that are not significantly different in terms of classification accuracy. Thus we conclude that PemNN is significantly more accurate than all baseline models. Among the baselines, LSTMFCN has the lowest rank, with its performance statistically similar to ResNet and FCN, but significantly different from Triplet and InceptionTime.

6.3. Functionality of two branches

This section evaluates the functionality of the force trace and physics-based branches in PemNN using simulated datasets with known ground truths.

Physics-based branch functionality The importance of the physics-based branch, which integrates polymer elastic

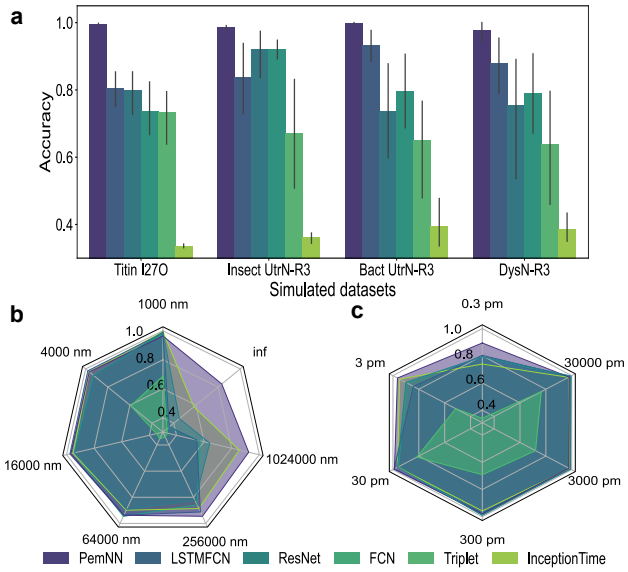

Figure 5. a) The average accuracy of all models across the four simulated datasets, with error bars indicating the standard deviations over five runs. b) The average accuracy across four simulated datasets over five runs as the contour length threshold varies, depicted in a radar chart where longer radii indicate higher accuracy. c) The average accuracy across four simulated datasets over five runs as the persistence length changes.

models, was evaluated by comparing PemNN (using both force trace and physics-based branches) to baselines that only utilized the force trace branch. As shown in Figure 5a, PemNN achieves near perfect accuracy (98.9$\pm$ 1.9 %) across four simulated datasets, significantly outperforming the baseline. The largest performance gap is in the Titin I27O simulated dataset, where PemNN outperformed baselines by 15%; the smallest gap occurs in the insect UtrN-R3 dataset, with PemNN maintaining a 6% higher accuracy.

Force trace branch Functionality Baselines were provided with the same input data as the physics-based branch to assess the contribution of the force trace branch. This was analyzed under two scenarios: one is changing contour length threshold in the filter, the other one is varying the persistence length L_p in the WLC model. Radar charts were employed to visualize model performance, with models represented by different transparent colors and higher radii indicating better accuracy. In Figure 5b, the contour length threshold is changed from 1,000 to 1,024,000 nm in multiples of four, with "inf" denoting no threshold. PemNN maintained near-perfect accuracy for thresholds between 1,000 and 64,000 nm, a range in which some baselines, such as LSTMFCN and ResNet, also perform well. However, as the threshold increases, baseline performance dropped sharply. In contrast, PemNN maintains robust performance, outperforming the

baselines by at least 7% at a threshold of 1,024,000 nm, with this margin growing to 25% when no threshold was applied. In Figure 5c, the persistence length L_p is varied logarithmically from 0.3 to 30,000 pm. Both PemNN and baselines achieved near-perfect accuracy for L_p values between 30 and 30,000 pm. However, at $L_p = 0.3$ pm and $L_p = 3$ pm, baseline performance dropped significantly, whereas PemNN maintained at least 3% and 10% higher accuracy, respectively. A more detailed discussion is provided in Appendix D.3. In conclusion, the physics-based branch enhances performance by incorporating polymer elastic models; the force trace branch increases robustness, ensuring reliable classification even when physics models are corrupted by parameter errors. By combining the strengths of both branches, PemNN consistently outperformed baselines in SMFS classification tasks, demonstrating its effectiveness under various conditions.

6.4. Performance with limited data

In this section, we focus on experimental datasets and analyze model performance under limited training data. The testing data is fixed at 80% of the entire dataset, while the deep learning models are trained using varying proportions of the training dataset. When the training proportion ranges from 5% to 20% (approximately 30 to 120 samples), PemNN consistently outperforms baselines, achieving accuracies of 79.6% and 85.2% at training proportions of 5% and 20%, respectively (Figure 6). Among the baselines, LSTMFCN performs the best but remains 11.4% and 2.9% less accuracy than PemNN at training proportions of 5% and 20%, respectively.

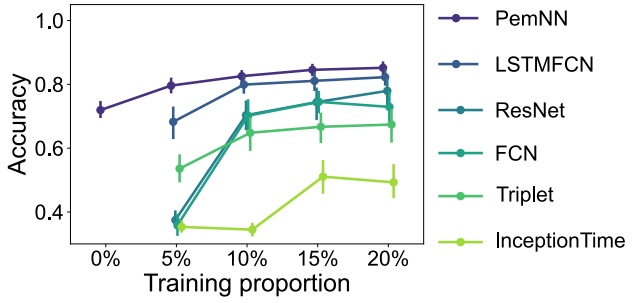

Figure 6. The performance of models trained with different proportions of experimental datasets (training proportion), with error bars indicating standard deviations across all experimental datasets over five runs.

At a training proportion of 0%, no experimental data is used during the training. Instead, a physics-guided pretraining strategy (see Appendix E) is used to eliminate the need for experimental labeling and mitigate human bias. In this approach, deep learning models are pre-trained on simulated datasets generated using physics-based models and subsequently evaluating them on corresponding experimental datasets. The pretraining strategy achieves comparable performance to models directly trained on experimental data, with an average accuracy of 72.0%, despite not utilizing any experimental data during its training. In our physics-based protein molecule unfolding model, we assume every protein domain behaves identically. However, many protein molecules, including utrophin and dystrophin have folded domains that are significantly different from each other. Despite being trained on simulation data using a single double-well potential model for the domains, PemNN demonstrates the capability to classify the number of protein molecules involved in experiments with protein molecules exhibiting heterogeneous domains (See Appendix E for details).

6.5. Application to SMFS data analysis

Here, we apply both machine learning and non-machine learning methods to Titin I27O data collected from a one-day experiment ($\sim$ 3000 curves) (See Appendix D.5 for utrophin and dystrophin). Titin I27O is a well-calibrated protein molecule with a most probable unfolding force of 204 ± 26 pN. In Figure 7, *RawData* includes all unfolding events. The *Heuristic* method, a non-machine learning approach, filters data with the WLC model (Rajaganapathy et al., 2019; Ramirez et al., 2023) that requires manual adjustments by experts to fine-tune parameters. *PemNN* was trained on our Titin I27O dataset and subsequently tested on the raw data to extract curves originating from single molecule. Notably, *PemNN* takes less than an hour to compute statistics, while the *Heuristic* method takes several hours, and *RawData* can take up to a full day.

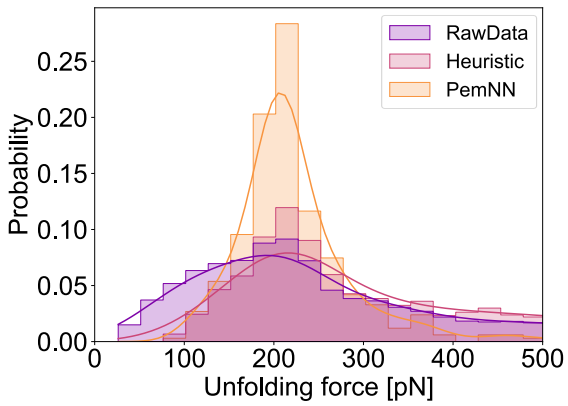

Figure 7. Application to SMFS data analysis of Titin I27O: *RawData* as the Baseline Method, *Heuristic* as non-machine learning method, and *PemNN* as the our proposed model.

PemNN achieves a most probable unfolding force of 206.68 pN, closely aligning with the expected value (204 ± 26 pN).

In contrast, the *Heuristic* and *RawData* yield most probable unfolding forces of 217.41 pN and 192.21 pN, respectively (Table 5). Furthermore, the inclusion of data not originating from single-molecule events resulted in broader force distributions. We quantified the sharpness of these distributions using the interquartile range (IQR), as listed in Table 5. Our method, *PemNN*, achieved an IQR of 52.63, which is only a quarter of the IQR observed from the *Heuristic* or the *RawData*, effectively filtering out confounding factors. These results highlight that *PemNN* effectively analyzes AFM data by accurately capturing key statistical features while effectively filtering out confounding factors from multiple molecules.

7. Conclusions

Single-molecule force spectroscopy (SMFS) data of protein molecules are time and resource intensive to collect. Currently, manual visual inspection remains the primary method for classifying force curves resulting from single molecules. This process typically requires a full day per experimental iteration of a specific molecule, with multiple iterations being standard practice in the field. These factors make it challenging to obtain precise statistics of single molecular force curves and to generate a large, annotated dataset appropriate for training deep learning models.

We developed a novel deep learning model that fuses a physics model based branch with another that does not employ physics to efficiently classify SMFS data as originating from no molecules, single molecules, or multiple molecules. We also applied state-of-the-art machine learning models (including ResNet, FCN, InceptionTime, and LSTMFCN) to SMFS data. Experimental datasets, obtained from four molecules (Titin I27O, bact UtrN-R3, insect UtrN-R3, and DysN-R3) are used to test and train the deep learning model. A Monte-Carlo engine, based on the physics of proteins, is developed and employed to provide simulated data. The presented approach achieves superior performance compared to state-of-the-art baseline methods on all simulated or experimental datasets. Remarkably, strong performance on experimental data even when trained solely on simulated data is corroborated. Furthermore, the model surpasses non-machine learning approaches in SMFS data analysis, demonstrating its effectiveness and reliability, while reducing processing time from a day to under an hour.

Acknowledgements

This project was supported by funding from NIH (5R01AR042423).

Impact Statement

Our deep learning model automates the classification of single molecule force spectroscopy (SMFS) data, effectively filtering out non-admissible data from experiments involving multiple molecules (as opposed to single molecule) without the need for expert knowledge. It also reduces the reliance on manual inspection, paving the way for faster and more consistent analysis of SMFS data. We expect the impact to be considerable on SMFS related research wherein laborious and tedious visual inspection can be automated. Additionally, this work provides SMFS experimental datasets from four important protein molecules, crucial to many biological pathways, which are made publicly available to support further research.

References

Ashkin, A., Dziedzic, J. M., Bjorkholm, J. E., and Chu, S. Observation of a single-beam gradient force optical trap for dielectric particles. *Optics letters*, 11(5):288–290, 1986.

Bagnall, A., Lines, J., Bostrom, A., Large, J., and Keogh, E. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, 31(3):606–660, 2017.

Binnig, G., Quate, C. F., and Gerber, C. Atomic Force Microscope. *Physical Review Letters*, 56(9):930–933, 1986.

Bornschlögl, T. and Rief, M. Single-Molecule Protein Unfolding and Refolding Using Atomic Force Microscopy. In Peterman, E. J. G. and Wuite, G. J. L. (eds.), *Single Molecule Analysis*, volume 783, pp. 233–250. Springer, 2011.

Bustamante, C., Marko, J. F., Siggia, E. D., and Smith, S. Entropic Elasticity of -Phage DNA. *Science*, 265(5178): 1599–1600, 1994.

Chen, Y., Xie, H., and Shin, H. Multi-layer fusion techniques using a CNN for multispectral pedestrian detection. *IET Computer Vision*, 12(8):1179–1187, 2018.

Dau, H. A., Bagnall, A., Kamgar, K., Yeh, C.-C. M., Zhu, Y., Gharghabi, S., Ratanamahatana, C. A., and Keogh, E. The UCR time series archive. *IEEE/CAA Journal of Automatica Sinica*, 6(6):1293–1305, 2019.

Demšar, J. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, 7 (Jan):1–30, 2006.

Doffini, V., Liu, H., Liu, Z., and Nash, M. A. Iterative Machine Learning for Classification and Discovery of

Single-Molecule Unfolding Trajectories from Force Spectroscopy Data. *Nano Letters*, 23(22):10406–10413, 2023.

Dudko, O. K., Hummer, G., and Szabo, A. Intrinsic rates and activation free energies from single-molecule pulling experiments. *Physical Review Letters*, 96(10):108101, 2006.

Dudko, O. K., Hummer, G., and Szabo, A. Theory, analysis, and interpretation of single-molecule force spectroscopy experiments. *Proceedings of the National Academy of Sciences USA*, 105(41):15755–15760, 2008.

Ervasti, J. M. Dystrophin, its interactions with other proteins, and implications for muscular dystrophy. *Biochimica et Biophysica Acta (BBA)-Molecular Basis of Disease*, 1772(2):108–117, 2007.

Fawaz, H. I., Forestier, G., Weber, J., Idoumghar, L., and Muller, P.-A. Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*, 33(4): 917–963, 2019.

Foumani, S. N. M., Tan, C. W., and Salehi, M. Disjoint-cnn for multivariate time series classification. In *2021 International Conference on Data Mining Workshops (ICDMW)*, pp. 760–769. IEEE, 2021.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hervás, R., Oroz, J., Galera-Prat, A., Goñi, O., Valbuena, A., Vera, A. M., Gómez-Sicilia, , Losada-Urzáiz, F., Uversky, V. N., Menéndez, M., Laurents, D. V., Bruix, M., and Carrión-Vázquez, M. Common Features at the Start of the Neurodegeneration Cascade. *PLoS Biology*, 10(5): e1001335, 2012.

Hewamalage, H., Bergmeir, C., and Bandara, K. Recurrent Neural Networks for Time Series Forecasting: Current Status and Future Directions. *International Journal of Forecasting*, 37(1):388–427, 2021.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Hoffer, E. and Ailon, N. Deep metric learning using triplet network. In *Similarity-based pattern recognition: third international workshop, SIMBAD 2015, Copenhagen, Denmark, October 12-14, 2015. Proceedings 3*, pp. 84–92. Springer, 2015.

Hua, C., Slick, R. A., Vavra, J., Muretta, J. M., Ervasti, J. M., and Salapaka, M. V. Two operational modes of atomic force microscopy reveal similar mechanical properties for homologous regions of dystrophin and utrophin. *bioRxiv*, 2024.

Huang, L., Zhang, C., and Zhang, H. Self-adaptive training: beyond empirical risk minimization. *Advances in neural information processing systems*, 33:19365–19376, 2020.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456. pmlr, 2015.

Ismail Fawaz, H., Lucas, B., Forestier, G., Pelletier, C., Schmidt, D. F., Weber, J., Webb, G. I., Idoumghar, L., Muller, P.-A., and Petitjean, F. InceptionTime: Finding AlexNet for time series classification. *Data Mining and Knowledge Discovery*, 34(6):1936–1962, 2020.

Israelachvili, J. N. *Intermolecular and surface forces*. Academic press, 2011.

Jobst, M. A., Schoeler, C., Malinowska, K., and Nash, M. A. Investigating Receptor-ligand Systems of the Cellulosome with AFM-based Single-molecule Force Spectroscopy. *Journal of Visualized Experiments*, (82):50950, 2013.

Karim, F., Majumdar, S., Darabi, H., and Chen, S. LSTM Fully Convolutional Networks for Time Series Classification. *IEEE Access*, 6:1662–1669, 2018.

Karim, F., Majumdar, S., Darabi, H., and Harford, S. Multivariate LSTM-FCNs for time series classification. *Neural Networks*, 116:237–245, 2019.

King, W. T., Su, M., and Yang, G. Monte Carlo simulation of mechanical unfolding of proteins based on a simple two-state model. *International Journal of Biological Macromolecules*, 46(2):159–166, 2010.

Kingma, D. P. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Leite, F. L., Bueno, C. C., Da Róz, A. L., Ziemath, E. C., and Oliveira, O. N. Theoretical Models for Surface Forces and Adhesion and Their Measurement Using Atomic Force Microscopy. *International Journal of Molecular Sciences*, 13(12):12773–12856, 2012.

Liu, M., Ren, S., Ma, S., Jiao, J., Chen, Y., Wang, Z., and Song, W. Gated transformer networks for multivariate time series classification. *arXiv preprint arXiv:2103.14438*, 2021.

Liu, Z., Liu, H., Vera, A. M., Bernardi, R. C., Tinnefeld, P., and Nash, M. A. High force catch bond mechanism of bacterial adhesion in the human gut. *Nature Communications*, 11(1):4321, 2020.

Livadaru, L., Netz, R. R., and Kreuzer, H. J. Stretching Response of Discrete Semiflexible Polymers. *Macromolecules*, 36(10):3732–3744, 2003.

Lyubchenko, Y. L. *Nanoscale Imaging: Methods and Protocols*. Springer, 2018.

Löning, M., Bagnall, A., Ganesh, S., Kazakov, V., Lines, J., and Király, F. J. sktime: A unified interface for machine learning with time series. *arXiv preprint arXiv:1909.07872*, 2019.

Mendell, J. R., Shilling, C., Leslie, N. D., Flanigan, K. M., al-Dahhak, R., Gastier-Foster, J., Kneile, K., Dunn, D. M., Duval, B., and Aoyagi, A. Evidence-based path to newborn screening for Duchenne muscular dystrophy. *Annals of neurology*, 71(3):304–313, 2012.

Nair, V. and Hinton, G. E. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 807–814, 2010.

Oberhauser, A. F., Marszalek, P. E., Erickson, H. P., and Fernandez, J. M. The molecular elasticity of the extracellular matrix protein tenascin. *Nature*, 393(6681):181–185, 1998.

Oberhauser, A. F., Hansma, P. K., Carrion-Vazquez, M., and Fernandez, J. M. Stepwise unfolding of titin under force-clamp atomic force microscopy. *Proceedings of the National Academy of Sciences*, 98(2):468–472, 2001. ISBN: 0027-8424 Publisher: The National Academy of Sciences.

Ortiz, C. and Hadziioannou, G. Entropic Elasticity of Single Polymer Chains of Poly(methacrylic acid) Measured by Atomic Force Microscopy. *Macromolecules*, 32(3):780–787, 1999.

Otten, M., Ott, W., Jobst, M. A., Milles, L. F., Verdorfer, T., Pippig, D. A., Nash, M. A., and Gaub, H. E. From genes to protein mechanics on a chip. *Nature Methods*, 11(11):1127–1130, 2014.

Pham, T.-A., Lee, J.-H., and Park, C.-S. MST-VAE: Multi-Scale Temporal Variational Autoencoder for Anomaly Detection in Multivariate Time Series. *Applied Sciences*, 12(19):10078, 2022.

Puchner, E. M., Franzen, G., Gautel, M., and Gaub, H. E. Comparing Proteins by Their Unfolding Pattern. *Biophysical Journal*, 95(1):426–434, 2008.

Rajaganapathy, S., McCourt, J. L., Ghosal, S., Lindsay, A., McCourt, P. M., Lowe, D. A., Ervasti, J. M., and Salapaka, M. V. Distinct mechanical properties in homologous spectrin-like repeats of utrophin. *Sci. Rep.*, 9(1):5210, 2019.

Ramirez, M. P., Rajaganapathy, S., Hagerty, A. R., Hua, C., Baxter, G. C., Vavra, J., Gordon, W. R., Muretta, J. M., Salapaka, M. V., and Ervasti, J. M. Phosphorylation alters the mechanical stiffness of a model fragment of the dystrophin homologue utrophin. *Journal of Biological Chemistry*, 299(2):102847, 2023.

Rief, M., Gautel, M., Oesterhelt, F., Fernandez, J. M., and Gaub, H. E. Reversible Unfolding of Individual Titin Immunoglobulin Domains by AFM. *Science*, 276(5315):1109–1112, 1997.

Rief, M., Gautel, M., Schemmel, A., and Gaub, H. E. The Mechanical Stability of Immunoglobulin and Fibronectin III Domains in the Muscle Protein Titin Measured by Atomic Force Microscopy. *Biophysical Journal*, 75(6):3008–3014, 1998.

Rief, M., Pascual, J., Saraste, M., and Gaub, H. E. Single molecule force spectroscopy of spectrin repeats: low unfolding forces in helix bundles. *Journal of Molecular Biology*, 286(2):553–561, 1999.

Schoeler, C., Malinowska, K. H., Bernardi, R. C., Milles, L. F., Jobst, M. A., Durner, E., Ott, W., Fried, D. B., Bayer, E. A., Schulten, K., Gaub, H. E., and Nash, M. A. Ultrastable cellulosome-adhesion complex tightens under load. *Nature Communications*, 5(1):5635, 2014.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Summers, C. and Dinneen, M. J. Improved mixed-example data augmentation. In *2019 IEEE winter conference on applications of computer vision (WACV)*, pp. 1262–1270. IEEE, 2019. ISBN 1-7281-1975-8.

Sundermeyer, M., Schlüter, R., and Ney, H. LSTM neural networks for language modeling. In *Interspeech 2012*, pp. 194–197. ISCA, 2012.

Tavenard, R., Faouzi, J., Vandewiele, G., Divo, F., Androz, G., Holtz, C., Payne, M., Yurchak, R., Rußwurm, M., Kolar, K., and Woods, E. Tslearn, A Machine Learning Toolkit for Time Series Data. 2020.

Tokozume, Y., Ushiku, Y., and Harada, T. Learning from between-class examples for deep sound recognition. *arXiv preprint arXiv:1711.10282*, 2017.

Tokozume, Y., Ushiku, Y., and Harada, T. Between-Class Learning for Image Classification. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5486–5494, Salt Lake City, UT, 2018. IEEE.

Waite, J. R., Tan, S. Y., Saha, H., Sarkar, S., and Sarkar, A. Few-shot deep learning for AFM force curve characterization of single-molecule interactions. *Patterns*, 4(1):100672, 2023.

Wang, L., Zhang, J., Liu, Y., Mi, J., and Zhang, J. Multimodal Medical Image Fusion Based on Gabor Representation Combination of Multi-CNN and Fuzzy Neural Network. *IEEE Access*, 9:67634–67647, 2021.

Wang, Z., Yan, W., and Oates, T. Time series classification from scratch with deep neural networks: A strong baseline. In *2017 International joint conference on neural networks (IJCNN)*, pp. 1578–1585. IEEE, 2017.

Yang, B., Liu, H., Liu, Z., Doenen, R., and Nash, M. A. Influence of Fluorination on Single-Molecule Unfolding and Rupture Pathways of a Mechanostable Protein Adhesion Complex. *Nano Letters*, 20(12):8940–8950, 2020a.

Yang, B., Liu, Z., Liu, H., and Nash, M. A. Next Generation Methods for Single-Molecule Force Spectroscopy on Polyproteins and Receptor-Ligand Complexes. *Frontiers in Molecular Biosciences*, 7:85, 2020b.

Zhang, S., Qian, H., Liu, Z., Ju, H., Lu, Z., Zhang, H., Chi, L., and Cui, S. Towards Unveiling the Exact Molecular Structure of Amorphous Red Phosphorus by Single-Molecule Studies. *Angewandte Chemie International Edition*, 58(6):1659–1663, 2019.

Zhang, X., Gao, Y., Lin, J., and Lu, C.-T. TapNet: Multivariate Time Series Classification with Attentional Prototypical Network. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):6845–6852, 2020.

Zheng, Y., Liu, Q., Chen, E., Ge, Y., and Zhao, J. L. Exploiting multi-channels deep convolutional neural networks for multivariate time series classification. *Frontiers of Computer Science*, 10(1):96–112, 2016.

Zimmermann, J. L., Nicolaus, T., Neuert, G., and Blank, K. Thiol-based, site-specific and covalent immobilization of biomolecules for single-molecule experiments. *Nature Protocols*, 5(6):975–985, 2010.

A. Simulating protein unfolding

Monte Carlo Simulation based methods are used widely in the SMFS studies, yielding results that closely align with experimental data (Liu et al., 2020; King et al., 2010). However, prior simulation frameworks are restricted to the idealized case of force curves arising from single molecule. To build a comprehensive training dataset, we incorporate the real cases of force curves generated by no molecule and multiple molecules. Additionally, we model the cases where only partial sections of molecules are present as well as the stochastic adhesion and detachment events, of the cantilever to the protein, to better approximate experimental data.

Algorithm 1 Monte Carlo Simulation

Require: v, $L_c^{(i)}$, $L_p^{(i)}$, $\Delta L_c^{(i)}$, $\Delta L_p^{(i)}$, k_0, $\Delta x^{\ddagger}$, $\Delta G^{\ddagger}$, N, $D^{(i)}$
Initialization: $z \leftarrow 0$, $U^{(i)} \leftarrow 0$
for $t \leftarrow 0 : \Delta t : T$ **do**
  $z \leftarrow z + v\Delta t$
  Solve (3) for extension x
  for $i \leftarrow 1 : 1 : N$ **do**
    Calculate $F_{WLC}^{(i)}$ using (4)
    Compute $k_{off}(F_{WLC}^{(i)})$ using (13)
    Compute $P_u^{(i)}(F_{WLC}^{(i)})$ using (5)
    Draw $\eta^{(i)} \sim \mathcal{U}_{[0,1]}$
    if $\eta^{(i)} < P_u^{(i)}(F_{WLC}^{(i)})$ **then**
      $L_c^{(i)} \leftarrow L_c^{(i)} + \Delta L_c$, $L_p^{(i)} \leftarrow L_p^{(i)} + \Delta L_p$
      $U^{(i)} \leftarrow U^{(i)} + 1$
    end if
    Draw $\eta_d^{(i)} \sim \mathcal{U}_{[0,1]}$
    if $U^{(i)} == D^{(i)}$ and $\eta_d^{(i)} < P_d(F_{WLC}^{(i)})$ **then**
      $L_c^{(i)} \leftarrow L_c^{(i)} + C_{Lc}$, $L_p^{(i)} \leftarrow L_p^{(i)} + C_{Lp}$
    end if
  end for
end for

For the simulations, N proteins are considered with i-th protein having $D^{(i)}$ folded domains attached between the substrate and the force probe. The base of the cantilever probe is moved away at a constant speed v; here the position of the base of the cantilever z is initialized at zero and is updated every Δt seconds. The protein extension x is determined by solving the equation,

$$k_c d = \sum_{i=1}^{N} F_{WLC}^{(i)}(x, L_c^{(i)}, L_p^{(i)}), \tag{3}$$

where k_c is the spring constant of the cantilever, and d is the deflection (Fig 1a). Here, $F_{WLC}^{(i)}(x, L_c^{(i)}, L_p^{(i)})$ is the worm-like chain (WLC) model that relates the force applied to the extension of i-th protein (See Appendix A.1), given by (Rief et al., 1999),

$$F_{WLC}^{(i)}(x, L_c^{(i)}, L_p^{(i)}) := \frac{k_B T}{L_p^{(i)}} \left[\frac{1}{4\left(1 - \frac{x}{L_c^{(i)}}\right)^2} - \frac{1}{4} + \frac{x}{L_c^{(i)}} \right], \tag{4}$$

where k_B is the Boltzmann constant, T is temperature, and $L_c^{(i)}$ and $L_p^{(i)}$ are the contour length and the persistence length of the i-th protein, respectively. For the i-th protein, the probability of a domain unfolding during the time interval Δt is found by

$$P_u^{(i)}(F_{WLC}^{(i)}) = (D^{(i)} - U^{(i)})(1 - e^{-k_{off}(F_{WLC}^{(i)})\Delta t}), \tag{5}$$

where $U^{(i)}$ is the number of unfolded domains which is initially set to zero, and $k_{off}(F_{WLC}^{(i)})$ is the transition rate that can be determined with the Dudko-Hummer-Szabo model (Dudko et al., 2008) (See Appendix A.2).

For determining unfolding events of i-th protein, a random number $\eta^{(i)}$ is generated uniformly from 0 to 1 and is compared to the unfolding probability $P_u^{(i)}(F_{WLC}^{(i)})$. No unfolding event is triggered if the random number is larger than $P_u^{(i)}(F_{WLC}^{(i)})$; the simulation will continue to the next time slot by adding time interval Δt. Otherwise, one of the domains is unfolded, leading to a increase in the number of unfolded domains, $U^{(i)}$, by 1, and the simulation continues to the next unfolding event if folded domains still exist after updating contour length and persistence length via adding increments $\Delta L_c^{(i)}$, $\Delta L_p^{(i)}$ respectively.

Once all domains in i-th protein unfold, the protein detaches from either the cantilever tip or substrate based on the detachment probability,

$$P_d(F) = \begin{cases} C_d & F \geq F_{td} \\ 0 & F < F_{td} \end{cases}, \tag{6}$$

where C_d is a constant and F_{td} is a random number sampled from the Gaussian distribution, denoting the threshold at which the detection of detachment begins. Upon detachment of the i-th protein, its WLC force is reduced to zero by adding large constants (C_{Lc} and C_{Lp}) to contour length $L_c^{(i)}$ and persistence length $L_p^{(i)}$ respectively. To better replicate experimental force curves, we introduce Gaussian noise to the WLC force immediately after its calculation at line 5 of Algorithm 1. The adhesion force (see Appendix A.3) is added at the end of the simulation process.

A.1. Polymer elastic models

Various polymer elasticity models have been developed to model the force-extension ($F - x$) behavior. The widely used models are the worm-like chain (WLC) model (Bustamante et al., 1994), the freely jointed chain (FJC) model (Ortiz & Hadziioannou, 1999), and the freely rotating chain (FRC) model (Livadaru et al., 2003).

The force-extension relationship for each model is given by the following equations:

- The WLC model:

$$F = \frac{k_B T}{L_p}\left[\frac{1}{4\left(1-\frac{x}{L_c}\right)^2} - \frac{1}{4} + \frac{x}{L_c}\right], \tag{7}$$

  where k_B is the Boltzmann constant, T is temperature, L_c is the contour length, and L_p is the persistence length.

- The FJC model:

$$\frac{x}{L_c} = coth(\frac{FL_k}{k_B T}) - \frac{k_B T}{FL_k}, \tag{8}$$

  where L_k is the Kuhn length.

- The FRC model:

$$\frac{x}{L_c} = 1 - \frac{k_B T}{2FL_b}. \tag{9}$$

  where L_b is the bond length.

In these models, the contour length L_c is a robust statistical parameter representing the maximal length of physically possible extension in a given folding state. (Puchner et al., 2008) first proposed solving for contour length L_c using each data point, rather than fitting each loading region with a polymer model. This approach has been widely adopted in SMFS studies (Liu et al., 2020; Yang et al., 2020b; Jobst et al., 2013; Otten et al., 2014; Zhang et al., 2019; Schoeler et al., 2014). Given the physically relevant constraints ($L_c > 0, x > 0, F > 0, x < L_c$) and fixed values for the persistence length L_p, Kuhn length L_k, and the bond length L_b, the contour length L_c can be calculated using the following equations for three polymer elasticity models:

- The WLC model (Jobst et al., 2013):

$$L_c^{WLC} = \frac{x}{6u}\left(3 + 4u + \frac{9 - 3u + 4u^2}{g(u)} + g(u)\right), \tag{10}$$

  where $u = \frac{FL_p}{k_B T}$ and $g(u) = \left(27 - \frac{27}{2}u + 36u^2 - 8u^3 + \frac{3\sqrt{3}}{2}\sqrt{-u^2\left[(4u-3)^3 - 108\right]}\right)^{1/3}$.

- The FJC model:

$$Lc^{FJC} = \frac{x}{coth(\frac{FL_k}{k_BT}) - \frac{k_BT}{FL_k}}. \tag{11}$$

- The FRC model, assuming bonds are connected by a fixed angle γ (Livadaru et al., 2003; Liu et al., 2020):

$$L_c^{FRC} = \begin{cases} \frac{3xk_BT}{Fa} & \text{for } \frac{FL_b}{k_BT} < \frac{L_b}{p} \\ \frac{x}{1-\left(\frac{4Fp}{k_BT}\right)^{-1/2}} & \text{for } \frac{L_b}{p} < \frac{FL_b}{k_BT} < \frac{p}{L_b}, \\ \frac{x}{1-\left(\frac{2FL_b}{k_BT}\right)^{-1}} & \text{for } \frac{p}{L_b} < \frac{FL_b}{k_BT} \end{cases} \tag{12}$$

where $a = L_b \frac{1+cos\gamma}{(1-cos\gamma)cos(\gamma/2)}$ is the Kuhn length and $p = L_b \frac{cos(\gamma/2)}{|ln(cos\gamma)|}$ is the persistence length.

A.2. The Dudko-Hummer-Szabo model

The Dudko-Hummer-Szabo (DHS) model (Dudko et al., 2008) provided an expression to find the force-dependent transition rate $k_{off}(F)$ by,

$$k_{off}(F) = k_0 \left(1 - \frac{\nu F \Delta x^\ddagger}{\Delta G^\ddagger}\right)^{\frac{1}{\nu}-1} e^{\beta \Delta G^\ddagger \left[1-\left(1-\frac{\nu F \Delta x^\ddagger}{\Delta G^\ddagger}\right)^{1/\nu}\right]}, \tag{13}$$

where k_0 is the intrinsic transition rate, $\Delta x^\ddagger$ is the distance to energy barrier, $\Delta G^\ddagger$ is the energy barrier height, and $\beta = \frac{1}{k_BT}$, and $\nu = 1/2$ or $2/3$, representing the cusp-like or linear-cubic energy landscape. Here k_0, $\Delta x^\ddagger$, and $\Delta G^\ddagger$ are defined in the absence of external force. The parameters for Titin I27O are reported by (Dudko et al., 2006), while those for UtrN-R3 and DysN-R3 are reported by (Hua et al., 2024).

Table 2. The DHS model parameters

MOLECULES	DHS $\nu = 1/2$		
	$ln(k_0)$	$\Delta x^\ddagger$ $[nm]$	$\Delta G^\ddagger$ $[k_BT]$
TITIN I27O (DUDKO ET AL., 2006)	-9.21	0.40	20.00
BACT UTRN-R3 (HUA ET AL., 2024)	-6.50	0.85	14.50
BACT DYSN-R3 (HUA ET AL., 2024)	-4.50	0.60	11.50
INSECT UTRN-R3 (HUA ET AL., 2024)	-2.50	0.41	9.80

A.3. Adhesion force model

The adhesive force can be composed of various components like van der Waals force, capillary force, and chemical forces, which depend on environmental conditions such as roughness, interacting angles, and wetness (Leite et al., 2012; Israelachvili, 2011). However, quantifying these environmental conditions is challenging, and they can vary significantly between experiments. Consequently, we adopt a straightforward yet versatile method to model adhesive force rather than a more intricate approach,

$$F_a(t) = \begin{cases} \frac{t}{t_1} F_{ad} & 0 < t < t_1 \\ \frac{t-t_2}{t_2-t_1} F_{ad} & t_1 < t < t_2 \\ 0 & else \end{cases}, \tag{14}$$

where the adhesive force increases linearly in the interval $[0, t_1]$, reaching the adhesive force threshold F_{ad} at t_1, then adhesion between the cantilever tip and the substrate begins to disconnect at t_1 and vanishes at t_2. The vanishing phase $[t_1, t_2]$ should be much faster than the adhesive phase $[0, t_1]$, with a common choice being to set t_2 close to t_1, for example, $t_2 = 1.1t_1$. To introduce stochasticity and enhance generality, we assume both F_{ad} and t_1 to be Gaussian distributed random variables with user-specified mean and standard deviation.

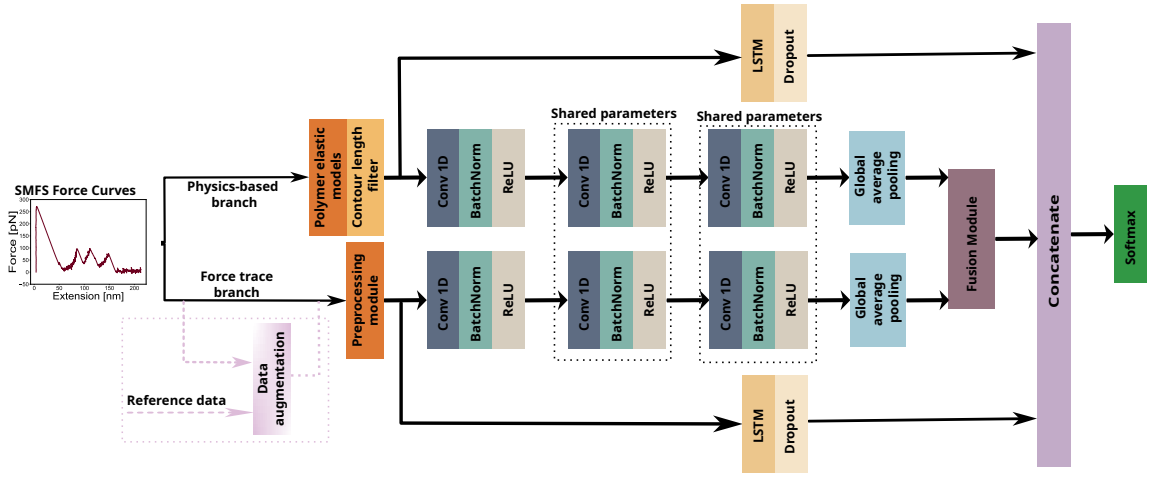

Figure 8. Overall architecture of the PemNN model with late fusion strategy. The data augmentation block is not enabled by default but is utilized through the physics-guided pretraining strategy described in Appendix E.

B. PemNN fusion methods

PemNN contains two branches: the force trace branch $\mathcal{T}$ and the physics-based branch $\mathcal{P}$. To uniquely identify a block, we use the notation $(\mathcal{C}, pos)$, where $\mathcal{C} \in \{\mathcal{P}, \mathcal{T}, \mathcal{PT}\}$ specifies the channel with $\mathcal{PT}$ representing the fusion of two branches, and $pos \in \{pre-module, CNN1, BN1, RELU1, CNN2, fusion, GAP, \dots\}$ indicates the name. For example, $(\mathcal{T}, CNN1)$ identifies the first convolutional layer in the force trace branch. We use $f_{\mathcal{C},pos}$ denotes the output of a block. PemNN supports two fusion strategies—late fusion and early fusion—which differ based on the dimensionality of the features being fused.

early fusion module is applied to fuse outputs of layers $(\mathcal{T}, RELU1)$ and $(\mathcal{P}, RELU1)$ (Figure 2). We discuss four different methods as follows:

- Early-Sum:
$$f_{Early-fusion}(\mathcal{F}^{(i)}, \mathcal{X}^{(i)}) = f_{\mathcal{T},RELU1}(\mathcal{F}^{(i)}) + f_{\mathcal{P},RELU1}(\mathcal{F}^{(i)}, \mathcal{X}^{(i)}) \tag{15}$$
which sums convolutional feature maps across branches.

- Early-Max:
$$f_{Early-fusion}(\mathcal{F}^{(i)}, \mathcal{X}^{(i)}) = \max\left\{f_{\mathcal{T},RELU1}(\mathcal{F}^{(i)}), f_{\mathcal{P},RELU1}(\mathcal{F}^{(i)}, \mathcal{X}^{(i)})\right\} \tag{16}$$
where the maximum value at each element is selected.

- Early-Wavg:
$$f_{Early-fusion}(\mathcal{F}^{(i)}, \mathcal{X}^{(i)}) = \alpha_{\mathcal{T}} f_{\mathcal{T},RELU1}(\mathcal{F}^{(i)}) + \alpha_{\mathcal{P}} f_{\mathcal{P},RELU2}(\mathcal{F}^{(i)}, \mathcal{X}^{(i)}) \qquad \text{s.t. } \alpha_{\mathcal{T}} + \alpha_{\mathcal{P}} = 1, \tag{17}$$
which computes a weighted sum of feature maps with coefficients $\alpha_{\mathcal{T}}, \alpha_{\mathcal{P}}$ learned during optimization.

- Early-Conv: The three methods described above require consistent output shapes for the layers $(\mathcal{T}, RELU1)$ and $(\mathcal{P}, RELU1)$. The (i, j) element of $f_{Early-fusion}(\mathcal{F}^{(i)}, \mathcal{X}^{(i)})$ is derived from the corresponding (i, j) elements of $f_{\mathcal{T},RELU1}(\mathcal{F}^{(i)})$ and $f_{\mathcal{P},RELU2}(\mathcal{F}^{(i)}, \mathcal{X}^{(i)})$, without any information exchange across different branches. To exchange information across branches, we propose using convolution with a set of filters $\omega^{Early-conv}$ of length of 1:
$$f_{Early-fusion}(\mathcal{F}^{(i)}, \mathcal{X}^{(i)}) = \left(f_{\mathcal{T},RELU1}(\mathcal{F}^{(i)}) \| f_{\mathcal{P},RELU1}(\mathcal{F}^{(i)}, \mathcal{X}^{(i)})\right) \circledast \omega^{Early-conv}$$
where $\circledast$ is the convolution operator, and the vertical concatenating operator $\|$ is defined as $v_i \| v_j = (v_{i0}, v_{i1}, \dots, v_{in}, v_{j0}, \dots v_{jn})^T$ given $v_i = (v_{i0}, v_{i1}, \dots, v_{in})^T$ and $v_j = (v_{j0}, v_{j1}, \dots, v_{jn})^T$.

Late-fusion strategy, where the fusion module is applied after the GAP layers, as depicted in Figure 8. Two methods are considered:

- Late-Concat:
$$f_{Late-fusion}(\mathcal{F}^{(i)}, \mathcal{X}^{(i)}) = f_{\mathcal{T},GAP}(\mathcal{F}^{(i)}) \| f_{\mathcal{P},GAP}(\mathcal{F}^{(i)}, \mathcal{X}^{(i)})$$
where GAP layers are concatenated directly.
- Late-Gating:
$$f_{Late-fusion}(\mathcal{F}^{(i)}, \mathcal{X}^{(i)}) = \left(g_{\mathcal{T}} f_{\mathcal{T},GAP}(\mathcal{F}^{(i)})\right) \| \left(g_{\mathcal{P}} f_{\mathcal{P},GAP}(\mathcal{F}^{(i)}), \mathcal{X}^{(i)})\right)$$
where GAP layers are concatenated with learnable weights. The gating weights $g_{\mathcal{T}}$,$g_{\mathcal{P}}$ are calculated as
$$(g_{\mathcal{T}}, g_{\mathcal{P}}) = \sigma\left(W \cdot \left(f_{\mathcal{T},GAP}(\mathcal{F}^{(i)}) \| f_{\mathcal{P},GAP}(\mathcal{F}^{(i)}, \mathcal{X}^{(i)})\right) + b\right),$$
where W and b are the weights and bias of a fully connected layer and $\sigma(\cdot)$ is softmax activation function.

In multivariate time series classification, the widely used approach is late fusion, which directly concatenates features extracted from Global Average Pooling (GAP) layers. For example, (Karim et al., 2019) concatenated the GAP layer of a convolutional block with output of an LSTM layer, (Zheng et al., 2016) concatenated GAP layers of convolutional blocks originated from different input branches, (Zhang et al., 2020) integrated both LSTM and GAP features from different branches, and (Liu et al., 2021) not only concatenated features from two transformer encoders but also introduced a gating mechanism to assign weights to these features. Unlike late fusion, early fusion focuses on combining convolutional feature maps from multiple branches. Examples includes (Wang et al., 2021), which employed an element-level maximum operator for feature fusion in image classification; (Chen et al., 2018), which summed feature maps for image detection; and (Pham et al., 2022), which concatenated feature maps extracted with different kernel sizes for anomaly detection.

C. Baselines and implementation

C.1. Baselines

Triplet The triplet network (Hoffer & Ailon, 2015) takes three samples: the anchor x, the positive sample x^+, and the negative sample x^-. Here, x is the sample under classification, x^+ comes from the same class as x, and x^- is of different class. All three samples are passed through the same network architecture, where the weights are shared, to learn representations in the embedding space. The triplet loss is employed,

$$L_{triplet} = max\{0, ||Net(x^+) - Net(x)||_2^2 - ||Net(x^-) - Net(x)||_2^2 + m\}, \tag{18}$$

where $||\cdot||$ denotes the L_2 norm, and m represents the margin parameter that controls the separation between positive and negative samples in the embedding space. The objective is to minimize the distance between the anchor and the positive sample while maximizing the distance between the anchor and the negative sample in the embedding space. Subsequently, the resulting embeddings are fed into the MLP described earlier to learn classification. The embedding network adopts the ResNet structure, which will be elaborated on below.

FCN Fully convolutional neural networks (FCN) (Wang et al., 2017) is composed of three convolutional blocks followed by a Global Average Pooling (GAP) layer and a softmax layer. Each convolutional block includes a 1D convolutional layer, batch normalization, and a Rectified Linear Unit (RELU) activation layer. The output of the final convolutional block is passed through a GAP layer, followed by a fully connected softmax layer. A stride of 1 with zero padding is used to preserve the length of time series data. The filter sizes and kernel sizes of the three convolutional layers are $\{128, 256, 128\}$ and $\{8, 5, 3\}$, respectively.

ResNet The ResNet (He et al., 2016) extends neural networks to very deep structures by incorporating shortcut connections. These connections enable the gradient flow directly through the network, easing the training of deeper models. Our Residual Network (ResNet) comprises three residual blocks followed by a GAP layer and a softmax layer. Each residual block contains three convolutional blocks similar to those in the FCN architecture. The output of the last convolutional block is added to the input of the residual block before proceeding to the next layer. The kernel size for the three convolutional layers in each residual block remains consistent with FCN architecture, set as $\{8, 5, 3\}$. The number of filters is the same within the same residual block, with filter sizes specified as $\{64, 128, 128\}$ for three residual blocks, respectively.

InceptionTime InceptionTime (Ismail Fawaz et al., 2020) ensembles the predictions of several InceptionNet models, each with different initializations, to overcome the high variance across different runs. The InceptionNet model consists of two residual blocks, followed by a global average pooling (GAP) layer and a softmax layer. Within each residual block are three Inception modules. Each Inception module begins with a bottleneck layer that reduces dimensionality while preserving the original sequence length through a kernel size and stride of 1. Subsequently, 1D convolutions with varying kernel sizes are applied to the bottleneck output, capturing patterns across different scales. Additionally, a max pooling layer, followed by another bottleneck layer, is applied to the original time series.

LSTMFCN LSTMFCN (Karim et al., 2019) employs the augmentation an LSTM layer and CNN layers along with a squeeze-and-excitation block to generate feature maps, followed by a softmax layer for time series classification. The three convolutional layers in LSTMFCN are configured with filter sizes of $\{128, 256, 128\}$ and kernel sizes of $\{8, 5, 3\}$, respectively. The LSTM layer has a hidden dimension of 8, followed by a dropout layer with a rate of 0.8.

C.2. Implementation details

In PemNN, the CNN blocks contains $\{128, 256, 128\}$ filters with kernel sizes of $\{8, 5, 3\}$, respectively; the LSTM layers are configured with a hidden dimension of 8, followed by a dropout layer with a rate of 0.8 to mitigate overfitting. The contour length threshold is set to 1000 nm, the persistence length $L_p = 300$ pm, Kuhn length $L_k = 300$ pm, and bond length $L_b = 150$ pm by default. The batch size is 16 for all models except for InceptionTime with 64, and the number of training epochs are kept at 200 epochs for all models. All models are trained with Adam (Kingma, 2014) with the learning rate 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 1e - 8$. The best-performing model, determined by achieving the lowest training loss, is then selected and evaluated on the testing data. These models are trained with Apple M1 Pro, which has 10-core CPU and 16-core GPU.

C.3. Data preprocessing

In both single molecule and multiple molecule cases, the force curve typically consists of two parts, separated by the detachment of the event. Our interest lies in the region before detachment, where unfolding events occur. Here, the force curves are trimmed to retain the region before detachment by identifying the detachment point. This trimming process is illustrated in the transition from Figure 9a to Figure 9b. In the force trace branch, data is resampled using linear interpolation to a default of 400 points, followed by min-max normalization, $x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$, where x_{min} and x_{max} are minimum and maximum values of the data x. This ensures the models focus on learning shapes rather than magnitudes, as shown in the transition from Figure 9b to Figure 9c. In the physics-based branch, 400 samples are extracted from the trimmed force curve by default, and the countour length L_c is calculated using polymer elastic models, illustrated in Figure 9d. We utilize tools from the publicly available time series library Tslearn (Tavenard et al., 2020) to implement the two preprocessing steps: resampling and normalization.

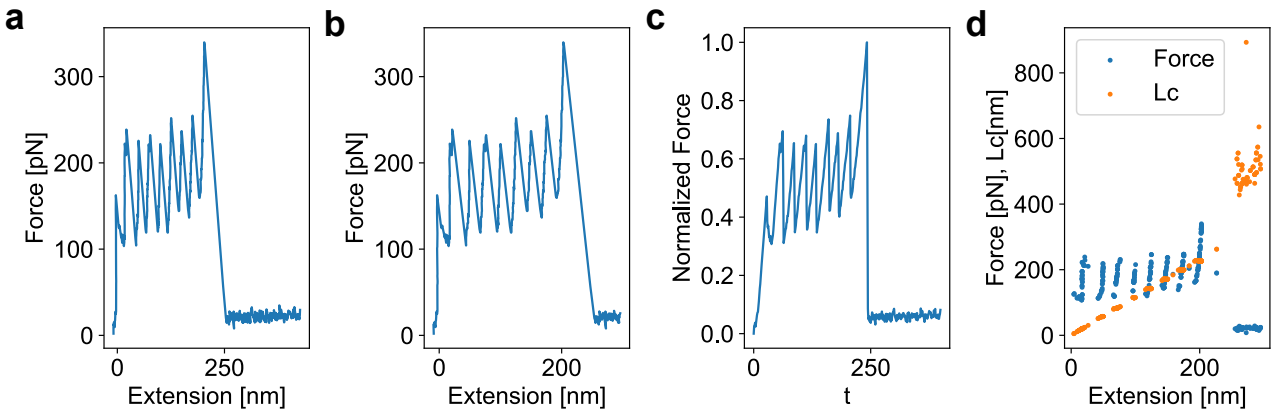

Figure 9. Data preprocessing example. (a) The original force curve. (b) The force curve after trimming. (c) The force data after normalization and resampling in the force trace branch. (d) The force curve with calculated L_c in the physics-based branch.

D. More Results

D.1. Results of the Discoidin Domain Receptors (DDRs) dataset

The **DDRs** dataset contains unfolding curves of interaction forces between DDRs and its ligand, collagen, measured by AFM at the single molecule level (Waite et al., 2023). The dataset contains three classes: 1) no molecule, 2) single molecule, and 3)multiple molecules. However, force curve collection was conducted at a single pulling speed, and the dataset is standardized to have equal lengths for each sample. However, our method, PemNN, is not applicable as extension data is not provided. Instead, we compare the performance of the remaining deep learning methods.

All deep learning models, except the Triplet model, were trained using 80% of the total dataset as the training dataset and tested on the remaining dataset. Their performance is compared to that of the triplet model reported in (Waite et al., 2023), as presented in Table 3. The dataset was pre-processed as proposed in (Waite et al., 2023), which involves applying a numerical first-order derivative, followed by a moving average filter with a window size of 13, and finally min-max normalization. Among the models, InceptionTime demonstrated the best performance, achieving approximately 20% higher overall accuracy and F1 score compared to the Triplet model.

Table 3. Overall accuracy, class accuracy, and F1-score of DDRs dataset for Triplet, MLP, FCN, ResNet, and InceptionTime are presented in the format of average (standard deviation) from 5 runs. Among these models, InceptionTime demonstrates superior performance compared to the others.

MODELS	OVERALL ACCURACY (%)	CLASS ACCURACY (%)			F1-SCORE(%)
		CLASS 0	CLASS 1	CLASS 2	
TRIPLET (WAITE ET AL., 2023)	66.70	73.30	63.30	63.30	66.70
FCN	77.10(5.15)	81.22(4.96)	77.49(13.59)	73.48(10.06)	77.30(5.32)
RESNET	79.03(4.84)	81.90(10.52)	83.97(4.74)	70.92(7.44)	79.02(4.88)
LSTMFCN	75.81(3.42)	70.91(2.49)	88.24(4.16)	71.30(9.02)	76.17(3.32)
INCEPTIONTIME	85.48(1.14)	97.14(2.61)	84.00(8.94)	72.50(12.96)	85.26(1.02)

D.2. Ablation study of model architecture

In this section, we explore the architectures of our model, PemNN, in greater detail to provide insights into optimizing model designs for SMFS classification. We examine 1) different fusion strategies, 2) the inclusion or exclusion of LSTM layers, and 3) different polymer elastic models.

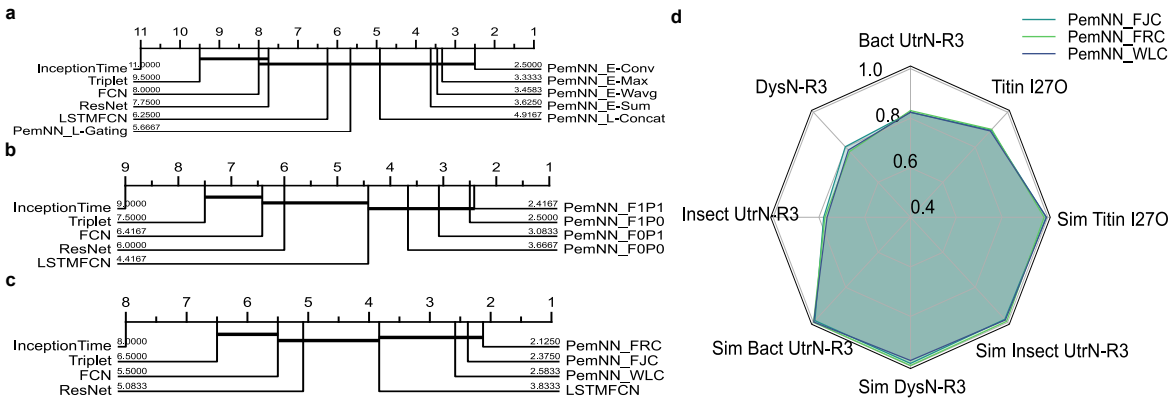

Figure 10. Ablation study of PemNN architecture. a) Critical diagram comparing various fusion strategies, where 'E-' and 'L-' denote early fusion and late fusion strategies, respectively. b) Critical diagram evaluating the incorporation of LSTM layers, with 'F1P1' indicating LSTM layers included in both the force trace and physics-based branches. c) Critical diagram of different polymer elastic models. d) Performance of PemNN across datasets with three distinct polymer elastic models.

Fusion Strategies Figure 10a presents results of various fusion strategies. Our model, PemNN, consistently outperforms the baselines regardless of the fusion strategy used. Moreover, all early fusion methods exhibit superior performance compared to late fusion methods. For late fusion methods, the GAP layers from the last convolutional blocks are concatenated, leading to significant information loss since each element in the GAP layer represents a single channel from the convolutional block. In contrast, early fusion methods directly integrate branches from first convolutional blocks, preserving more information. Among the early fusion strategies, the Early-Conv strategy stands out, likely due to its use of filters to effectively process information across branches.

LSTM Layers Figure 10b shows the impact of incorporating LSTM layers into the two branches of PemNN. Regardless of the specific use of LSTM layers, PemNN consistently outperforms the baselines, with all models showing statistically significant improvements over the baselines, except for LSTMFCN. Including LSTM layers in both the trace and physics-based branches achieves the best overall performance, with the average ranking of 2.4167.

Different polymer elastic models Regardless of the chosen polymer elastic model, PemNN consistently outperforms the baselines, as shown in Figure 10c. Furthermore, employing the FRC and FJC models improves performance compared to the WLC model. Figure 10d presents the performance of the three polymer elastic models across simulated and experimental datasets using a radar chart. Even though the WLC model is used to generate the simulated datasets, PemNN with the FRC and FJC models performs comparably, with differences in averaged accuracy remaining within 3%. For real experimental datasets, PemNN maintains this robustness, with accuracy differences across the three models limited to within 3% for all four protein molecules.

These results highlight the consistent performance of PemNN in classification tasks, regardless of fusion methods, the incorporation of LSTM layers, or the choice of Polymer elastic models, and its ability to consistently outperform baselines.

D.3. Closer look at force trace branch functionality

In Section 6.3, we demonstrate that our model, PemNN, is robust to variations in physical parameters, specifically the contour length threshold and persistence length. Here we provide a detailed discussion.

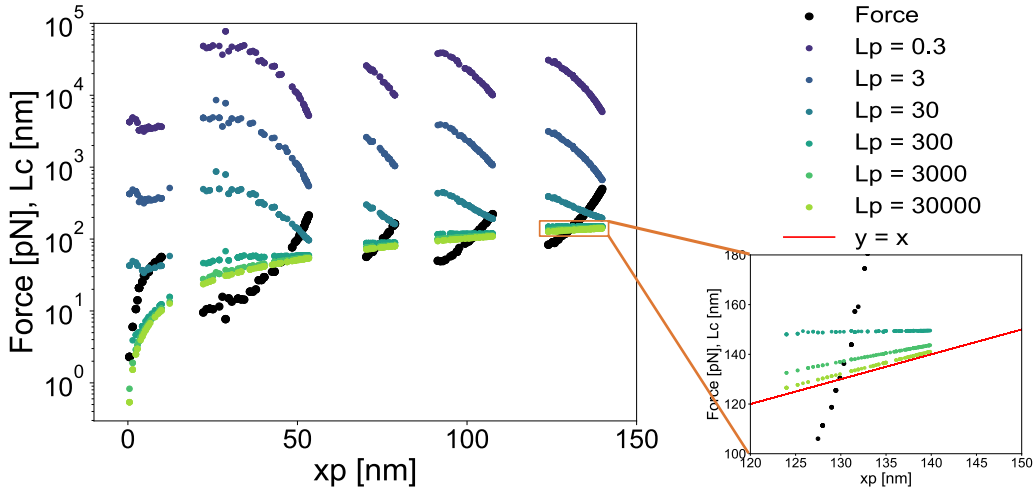

Figure 11. Contour length calculations using the WLC model with varying persistence lengths (L_p), visualized with a simulated force curve of Titin I27O.

Contour length threshold Contour lengths typically span the micrometer scale. However, factors such as signal noise or deviations from the polymer elastic models can lead to exaggerated contour length values, potentially degrading model performance. Despite these challenges, PemNN exhibits robust performance even when datasets include such extreme values.

Persistence length The persistence length L_p quantifies the bending stiffness of a polymer. When L_p is underestimated, the polymer appears excessively flexible, causing the contour length L_c to be overestimated. Conversely, when L_p is overestimated, the polymer appears stiffer, and L_c approaches the extension x. Figure 11 illustrates the relationship between L_p and L_c using the WLC model applied to a force curve from the Titin I27O dataset. At $L_p = 300$, contour length L_c remains consistent within unfolding events, indicating a correctly chosen L_p. Underestimated L_p (0.3-30) results in large and inconsistent L_c, leading to performance degradation. When L_p is overestimated (3000, 30000), L_c becomes linear with x, converging toward an identity function (red line in the inset of Figure 11).

D.4. Performance under varying train/test splits

We evaluated our model and baseline methods using a range of train/test split ratios on experimental datasets. Table 4 reports the mean accuracy (with standard deviation in parentheses) across all experimental datasets over five independent runs. Our model consistently outperforms baseline methods, particularly under low-data conditions, achieving at least 11.4% higher average accuracy when trained on only 5% of the data. As the training data increases, our approach maintains superior accuracy while exhibiting smaller standard deviations.

Table 4. Model performance on experimental datasets with varying train/test split ratios. Values represent mean accuracy (%) with standard deviations over five runs in parentheses.

TRAIN/TEST RATIO	5/80	40/60	60/40	80/20
PEMNN	79.61(5.24)	86.82(4.86)	88.50(5.09)	88.54(5.68)
LSTMFCN	68.26(12.12)	85.12(7.49)	85.51(6.87)	87.66(7.66)
RESNET	37.47(6.33)	81.01(9.13)	84.62(7.13)	84.47(9.26)
FCN	35.88(7.37)	75.50(8.92)	75.48(13.15)	79.43(10.18)
TRIPLET	55.58(10.17)	66.78(12.06)	69.15(12.26)	66.92(11.45)
INCEPTIONTIME	35.32(3.97)	80.78(12.78)	81.35(9.07)	84.76(11.58)

D.5. Application to SMFS data analysis

In Section 6.5, we examined the unfolding force distributions of Titin I27O using three methods: *RawData* as the baseline, *Heuristic* as a non-machine learning approach, and *PemNN*, our proposed model. Expanding on this analysis, we provide additional details for utrophin and dystrophin molecules, as illustrated in Figure 12.

Unfolding force distributions are visualized using violin plots, where black stars indicate the values with the highest probability. The violins depict data distributions with kernel density estimation (shown as black lines on either side), and the width of each curve represents the relative frequency of data points. For each protein molecule, Titin I27O, bact UtrN-R3, insect UtrN-R3, and DysN-R3, the unfolding distributions generated by *PemNN* are significantly more concentrated compared to those from *RawData* and *Heuristic*, as quantified by the interquartile range (IQR) listed in Table 5. Furthermore, the most probable forces obtained with *PemNN* differ from those achieved by the *Heuristic* method by no more than 10 pN. These results highlight that *PemNN* effectively analyzes AFM data by accurately capturing key statistical features while effectively filtering out confounding factors from multiple molecules.

Table 5. The most probable forces (in pN), along with the interquartile range (IQR) in parentheses, for all four protein molecules analyzed using different methods during AFM data analysis.

TITIN I27O			BACT UTRN-R3		
RAWDATA	HEURISTIC	PEMNN	RAWDATA	HEURISTIC	PEMNN
192.21(234.52)	217.41(270.46)	206.68(52.63)	60.50(63.22)	72.84(66.55)	70.40(36.97)
INSECT UTRN-R3			DYSN-R3		
RAWDATA	HEURISTIC	PEMNN	RAWDATA	HEURISTIC	PEMNN
81.41(85.29)	90.58(80.57)	84.66(39.98)	63.76(53.18)	74.38(53.10)	72.54(26.79)

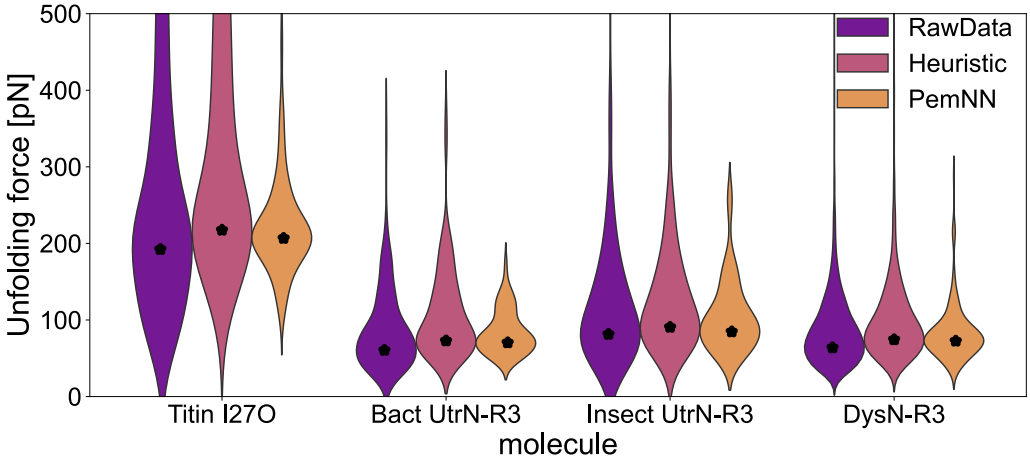

Figure 12. Application of PemNN to AFM Data Analysis: *RawData* as the Baseline Method, *Heuristic* as non machine learning method, and *PemNN* as the our proposed model. Unfolding force distributions are depicted using violin plots, where black stars signify the positions of values with the highest probability, with values detailed in Table 5. The violins illustrate data distributions using kernel density estimation as black lines on each side, with the width of each curve indicating the relative frequency of data points.

E. Physics-guided pretraining strategy

Given the challenge of constructing a large, well-annotated dataset using SMFS experimental data and to eliminate experimental labeling and human bias, we employed a physics-guided pretraining strategy. In this approach, deep learning models are pre-trained on simulated datasets generated using physics-based models and subsequently evaluating them on corresponding experimental datasets.

E.1. Pretraining deep learning models with the physics of protein unfolding

By utilizing simulation data, we effectively incorporate the underlying physics of protein unfolding into our analysis. Our simulation (Appendix A) is carried out using the WLC model. The WLC model encapsulates the physics of the protein unfolding, accurately describing the entropic spring like behavior of the protein between two unfolding events of the protein, which is also corroborated by experimental data. Subsequently, we test the performance of these pre-trained deep learning models using experimental data.

Data augmentation via linear combinations of examples from different classes is shown to be effective in image classification (Summers & Dinneen, 2019; Tokozume et al., 2018; Huang et al., 2020) and sound recognition (Tokozume et al., 2017). Here, we incorporate $M \in \mathbb{N}$ reference data into the force trace branch of PemNN to augment the force data under classification (Figure 8). The reference data are randomly sampled simulated force data from the training dataset. Each reference data $\mathcal{F}_j^{(i)}$ is augmented with the force data $\mathcal{F}^{(i)}$ via the difference $(\mathcal{F}^{(i)} - \mathcal{F}_j^{(i)})$ as additional channels, resulting in a total of $M+1$ channels. The force data $\mathcal{F}^{(i)}$ can be either a simulated or experimental force curve undergoing classification. This augmented input $\left[\mathcal{F}^{(i)}, \mathcal{F}^{(i)} - \mathcal{F}_1^{(i)}, \cdots, \mathcal{F}^{(i)} - \mathcal{F}_M^{(i)}\right]^T$, comprising the force data and the reference data, is then passed through the force trace branch.

E.2. Physics-guided pretraining strategy performance

We first evaluate the data augmentation block discussed in Appendix E.1 by empirically testing three methods for selecting reference data:

- random balanced: Reference data are randomly selected while maintaining an equal number of samples from each class for every input data.
- random unbalanced: Reference data are randomly selected without class balance for every input data.

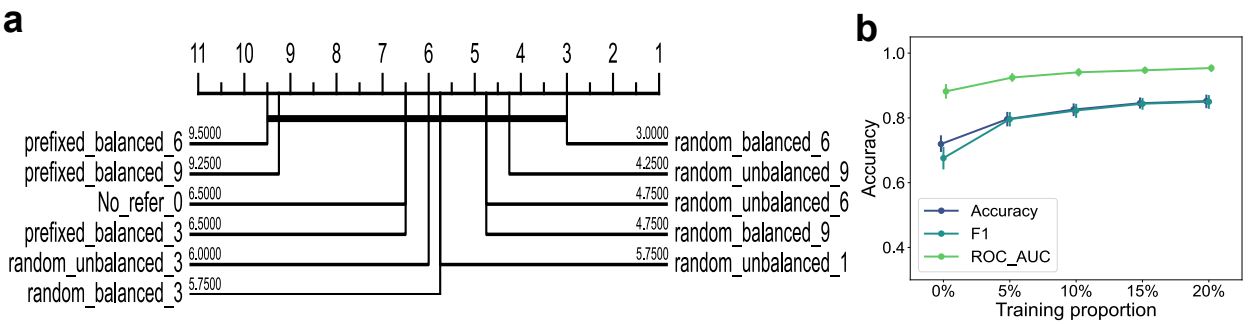

Figure 13. Pretraining strategy results with PemNN. a)Critical difference diagram comparing different data augmentation methods. b)The physics-guided pretraining strategy (train size = 0%) slightly underperforms compared to direct training with experimental data (train sizes = 5% to 20%).

- prefixed balanced: Reference data are fixed for all input data and are selected equally from each class.

Figure 13a presents the critical diagram comparing these selection criteria for different numbers of reference data M. Both the random balanced and random unbalanced criteria achieve higher rankings compared to the prefixed balanced criterion and the case where no reference data is used. Among these, the random balanced approach with $M = 6$ achieves the highest ranking and and is adopted for subsequent evaluations.

The physics-guided pretraining strategy demonstrates performance comparable to cases where the model is directly trained using experimental data, despite not utilizing any experimental data during its training. Specifically, the physics-guided pretraining strategy with PemNN achieves an average accuracy of 72.0% and an average ROC-AUC of 0.88 without invovling any experimental data (Figure 13). These metrics are only 13% and 0.07 lower, respectively, compared to when 20% of the experimental data is incorporated during training.

E.3. Similarities between simulated and experimental data

We compared unfolding forces from experimental and simulated data based on their most probable values and interquartile ranges (IQR) (see Table 6). Most probable values closely match, with differences from 1 pN (Titin I27O) to ∼10 pN (10%) (Bact UtrN-R3, Insect UtrN-R3, DysN-R3), indicating reasonable simulation accuracy. The IQR discrepancy likely arises from using a single double-well potential for domains in simulations. We note a key insight from our study; neural networks pre-trained on simulated data with homogeneous domains can effectively classify the number of proteins involved in experiments with heterogeneous protein domains.

Table 6. The most probable forces (in pN), along with the interquartile range (IQR) in parentheses, for both experimental and simulated data across all four protein.

	TITIN I27O	BACT UTRN-R3	INSECT UTRN-R3	DYSN-R3
EXPERIMENTAL	216.35(50.17)	81.58(64.07)	89.45(70.86)	91.25(65.64)
SIMULATED	217.75(36.17)	85.38(22.28)	96.65(40.20)	79.75(25.09)

E.4. Non-homogeneity introduces more challenges

The ROC curves (Figure 14) for PemNN were generated using the One-vs-Rest strategy, where a given class is regarded as the positive class and the remaining classes are regarded as the negative class as a bulk. For Titin I27O, the ROC-AUC remains consistently high, above 0.85 in all cases, regardless of whether the training data is experimental or simulated (Figure 14a, Table 7). For both utrophin and dystrophin, the no-molecule class consistently performs well whether experimental data is used for training or not. However, significant improvements are observed for the single-molecule and multiple-molecule classes when experimental data is used for training instead of simulation data, with ROC-AUC of single-molecule class

increasing by at least 0.2 for insect UtrN-R3, bact UtrN-R3, and DysN-RR3 (Figure 14b-d, Table 7). This suggests notable differences between simulation and experimental data for both utrophin and dystrophin.

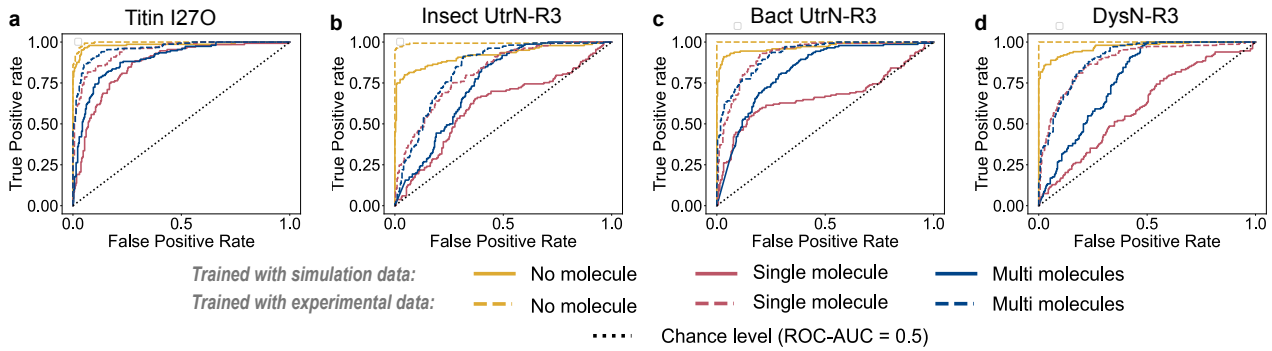

Figure 14. ROC curves from a single run for four datasets, (a) Titin I27O, (b) Insect UtrN-R3, (c) Bact UtrN-R3 and (d) DysN-R3 using PemNN. ROC curves are plotted using One-vs-Rest strategy, with no molecule, single molecule and multiple molecules classes depicted in yellow, red and blue, respectively. Solid lines represent results trained with simulation data, while dashed lines indicate results trained with experimental data. These ROC curves are generated using 80% of experimental data, with the corresponding ROC-AUC in Table 7.

The ROC curves allow us to adjust the optimal probability threshold. We can achieve high precision but accept low sensitivity to ensure the reliability of single molecule data statistics. The probability threshold can be chosen to tune for a minimum sensitivity or for a minimum specificity. For example, we have selected the thresholds to achieve an average of 93% of the positive identifications of single molecule are correct across four experimental datasets. Thus the method can effectively filter data such that inferences can be drawn from true single molecule experiments. Alternatively, we can aim for high sensitivity but accept low precision to avoid losing data, given the scarcity and high cost of producing SMFS data (Appendix E.5).

We further investigated the possible reasons for the superior performances of Titin I27O compared to insect UtrN-R3, bact UtrN-R3, and DysN-R3. Titin I27O has identical domains whereas utrophin and dystrophin have heterogeneous domains. The simulation model we use assumes identical folded domains in a protein. A model that can simulate a molecule with heterogeneous domains requires the model parameters for each domain. These parameters are not available for insect UtrN-R3, bact UtrN-R3, and DysN-R3. Thus, we hypothesize that when information about the molecular domains is missing in the simulation model, the usage of experimental data in training provides substantial improvements in performance. This is the case with molecules with heterogeneous domains.

Table 7. ROC-AUC of ROC curves in Figure 14.

	Trained with simulation data			Trained with experimental data		
	No molecule	Single molecule	Multiple molecules	No molecule	Single molecule	Multiple molecules
Titin I27O	0.98	0.86	0.90	1.00	0.95	0.96
Insect UtrN-R3	0.92	0.61	0.75	0.99	0.81	0.84
Bact UtrN-R3	0.97	0.67	0.82	1.00	0.92	0.93
DysN-R3	0.97	0.61	0.77	1.00	0.88	0.90

E.5. Adjusting probability thresholds

The primary goal is to identify single molecule force curves, so we simplify the problem from multi-class to binary classification, focusing on distinguishing force curves from single molecules versus those from no molecule and multiple molecules. By leveraging the ROC curve, which illustrates the true positive rates (TPR) against the false positive rates (FPR) by varying the probability threshold t_p, we can choose the optimal probability thresholds t_{op} to classify these binary classes. The optimal probability threshold t_{op} is determined by maximizing the difference between TPR and FPR with a weight $\alpha \in \mathbb{R}$ on FPR,

$$t_{op} = argmax_{t_p}(TPR - \alpha \cdot FPR). \tag{19}$$

Subsequently, optimal thresholds with varying α are applied to the binary classification task. The results, compared to the original threshold, are presented in Figure 15 for ResNet. The models are trained on simulation data and then evaluated on experimental data.

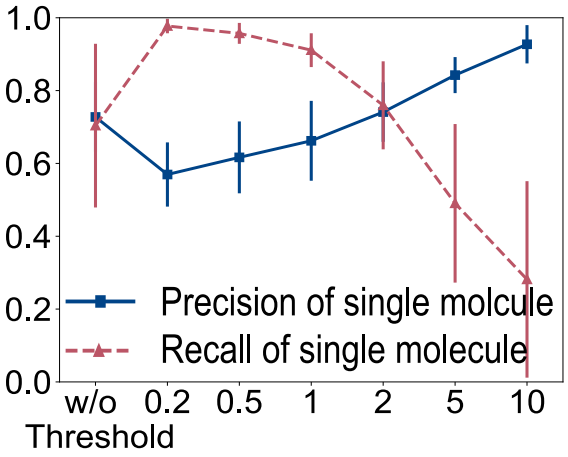

Figure 15. Performance of PemNN with varying probability thresholds for all four experimental datasets. The x-axis represents values of α, with 'w/o Threshold' indicating no threshold is used.

A larger weight α places greater emphasis on false positive, thereby increasing precision. For example, the precision of single molecule class improves to 0.93 when $\alpha = 10$. However, this improvement comes with a trade-off in recall. This adjustment enhances the reliability of single molecule data statistics, as we can be more confident that no-molecule and multiple-molecules force curves are not misclassified as single-molecule class. Conversely, if the preference is to accept some false positives rather than exclude any true positives as the data are expensive and scarce, $\alpha = 0.2$ would be the choice to achieve high recall, albeit with lower precision.

E.6. Discussion and limitations

A significant insight of the study is that we can perform the task of classification of data under no molecule, single molecule, and multiple molecules (for proteins with heterogenous domains) being pulled where the training is done based on simulation data, where a single double-well potential model of the domains is employed. This implies that there is enough other information on the force-curves that allows discrimination between no, single, and multiple proteins-based force-curves. We emphasize that neural networks trained on simulated data using homogeneous domains have the capability to classify between the number of proteins being involved in the experiments of protein molecules with heterogeneous domains.

We identify the following limitations for the physics-guided pretraining strategy. First, the simulation algorithm relies on energy landscape parameters, which are not straightforward to obtain and typically necessitate experiments involving multiple pulling speeds or constant forces. Transforming our physics-guided pretraining strategy to be agnostic to the energy landscape parameters is beyond the scope of this study. Second, in our physics-based protein unfolding model, we assume every protein domain behaves identically. However, many proteins, including utrophin and dystrophin have folded domains that are significantly different from each other. Developing methods that can extract the individual energy landscapes of the different dissimilar domains in a protein is likely to aid in identifying single molecule force curves arising from proteins with dissimilar domains. The development of techniques to extract distinct energy landscapes from a small amount of experimental data should be investigated in the future.