SIMULATION-BASED SENSOR-INVARIANT GRASP STABILITY PREDICTION MODEL

VANDAM DINH, EFI PSOMOPOULOU

Project Report submitted in support of the
degree of Master of Engineering

University of Bristol

22nd April 2026

© Vandam Dinh, 22nd April 2026
Simulation-Based Sensor-Invariant Grasp Stability Prediction Model

Though I know I'll never lose affection
For people and things that went before
I know I'll often stop and think about them
In my life, I love you more

In my life, I love you more

— John Lennon & Paul McCartney

ACKNOWLEDGMENTS

Thank you to my supervisor, Efi Psomopoulou, for guiding me throughout this project and keeping me on my toes! To Mum, Dad and my brother, Duydan, thank you for supporting me no matter what!

To my past colleagues at Dyson: the platform team – Luke Packman, David Gregan, Hibah Fadel and Nivesh Modi – thank you for introducing me to the world of software development, your patience and kindness have left such a positive mark on me.

The robotics simulation team – Ben Eggington, Harry Alcott, Tom Bath, Israel Becker Pilan, Lee Perry, Aidar Vasiliauskas, Francisco Sarmento, Oliver McIntosh and Bradley Copp – thank you for teaching me so much about robotics and allowing me to solve so many weird and wacky problems!

To my A-Level maths teachers, Mr Miles and Mrs Hunt, thank you for inspiring and supporting me in the classroom. I'm where I am now because you built up my confidence and made me fall in love with maths!

Finally, a huge shoutout to my friends: Julian Hall, thank you for always being there for me and grounding me with everything I do.

The project lab gang – Ciara Brown, Jack Buxton, Nesan Pathmakanthan and Joe Turner – thank you for all the laughs, snacks and making the wordles, crosswords and late nights oh so more enjoyable!

USE OF AI

Large language models (LLMs), such as ChatGPT, Claude and Gemini, have been used in this project for the following:

- To help me learn topics and understand implementation feasibility of different ideas.
- To assist with research, through tools such as Google Scholar Labs[1] and alphaXiv[2].
- To prototype and analyse code, through tools such as Codex[3].
- To assist with writing, specifically with suggesting structure, flow/readability improvements and to draft my ideas.
- To write LaTeX code, to help me develop this template.

CODE

The code used in this dissertation is available at: `https://github.com/vandamd/thesis`. Instructions to run are provided in the `README.md` file. You are free to do whatever you'd like with the code. I have used the MIT license.

1 https://scholar.google.co.uk/scholar_labs/search
2 https://www.alphaxiv.org/
3 https://chatgpt.com/codex

ABSTRACT

Humans judge grasp security through a combination of tactile, visual, and proprioceptive cues, with fingertip contact feedback playing a central role before and during lifting. Robotic hands, by contrast, still largely rely on reactive methods that respond only after signs of slip or instability have already appeared. This dissertation asks whether a robot can use a short recent history of contact information to warn that a grasp is likely to become unstable in the near future.

To answer this, the project develops a simulation-based pipeline in MuJoCo using the ORCA hand and a set of YCB objects. Candidate grasps are generated through energy-based optimisation, replayed under gravity, and recorded over time. From each replay, the system extracts compact fingertip contact features: normal and shear force at each fingertip. These force histories are then used to build a supervised prediction task in which the model must decide whether a grasp will remain stable or become unstable shortly afterwards. The study compares three model families for this task: a CNN, a GRU, and an LSTM.

Across unseen test objects, all three models learn useful predictive structure from these force histories alone. This shows that predictive grasp assessment does not necessarily require raw tactile images tied to one specific sensor design. On the baseline benchmark task, the GRU gives the strongest overall performance. However, when observation time is treated as part of the total decision latency, much shorter windows become more attractive, and a short-window LSTM becomes the preferred practical operating point under the latency-aware selection rule used.

CONTENTS

1 INTRODUCTION

The human hand performs a diverse range of complex actions, including grasping, holding, and manipulating objects. Designing an artificial hand that matches this level of control and adaptability remains a fundamental challenge [1]. Human grasp control does not depend on touch alone; it combines tactile, visual, and proprioceptive cues with predictions about the consequences of action. This anticipatory view is consistent with active-inference accounts, which treat perception and action as coupled processes shaped by beliefs about future states and uncertainty [2, 3]. Related motor-control work shows that people adapt their movements as uncertainty changes, rather than only correcting errors after they occur [4]. The present work focuses on the tactile component of this broader process. Once contact is made, fingertip information facilitates subtle adjustments to hand pose and grip force before grasp failure occurs [5]. In this sense, human grasping functions not merely as a reactive process, but as an anticipatory one.

Conversely, robotic hands typically lack this anticipatory ability, relying instead on reactive strategies. To bridge this gap, modern robotic systems increasingly integrate touch with vision. Vision-based tactile sensors such as GelSight [6], DIGIT [7], TacTip [8], and FingerVision [9] have improved manipulation performance across applications such as warehouse automation [10] and automated recycling [11]. Yet, researchers still commonly treat grasp stability as a reactive problem. Controllers attempt recovery by adjusting grip force or pose once they detect instability.

A grasp often fails through slip. Slip can begin as incipient slip, where only part of the contact interface starts to move, before progressing to gross slip, where the object slides relative to the hand [12]. For this reason, many reactive systems focus on detecting incipient slip rather than waiting for gross slip, as this enables earlier corrective action [13]. Detecting incipient slip still matters, but it remains reactive: some part of the contact has already started to move. For dexterous manipulation, a more ambitious goal involves recognising increasing grasp risk before any observable slip cue appears.

This dissertation examines grasp stability from a predictive perspective. The central question is whether short histories of contact information can provide an early warning of near-future grasp instability. Constructing such a model from real tactile data presents two main difficulties: first, collecting large, well-labelled temporal datasets on hardware takes significant time and expense; second, raw tactile observations often depend heavily upon the specific sensor employed. Thus, the present work adopts a simulation-based approach and focuses on physically meaningful contact features, such as fingertip normal and shear contact forces, rather than raw tactile images.

This dissertation makes three methodological contributions. First, it defines a simulation-based pipeline that combines dexterous grasp generation, controlled grasp replay, contact-feature extraction, and temporal window labelling for predictive grasp-stability learning. Secondly, it studies whether short histories of normal and shear contact force can provide useful early warning of instability without relying on sensor-specific tactile imagery. Thirdly, it establishes an evaluation strategy that compares temporal prediction models across training objects and unseen objects using the same prediction task and labelling scheme.

The working hypothesis is that short histories of normal and shear contact dynamics contain enough information to provide useful early warnings of grasp instability. If this hypothesis holds, predictive grasp assessment may not require raw sensor-specific tactile imagery, and lower-dimensional contact features may still support anticipatory manipulation.

This thesis is structured as follows. Chapter 2 reviews prior work on tactile sensing, slip detection, predictive grasp assessment, and simulation-based learning. Chapter 3 describes the proposed pipeline, grasp generation, data extraction, predictive window labelling, model design, and training protocol. Chapter 4 evaluates the predictive models and discusses what the results imply for model design and operating-window choice. Chapter 5 concludes the dissertation by returning to the central question, summarising the main contributions, and outlining the most important limitations and next steps.

2 LITERATURE REVIEW

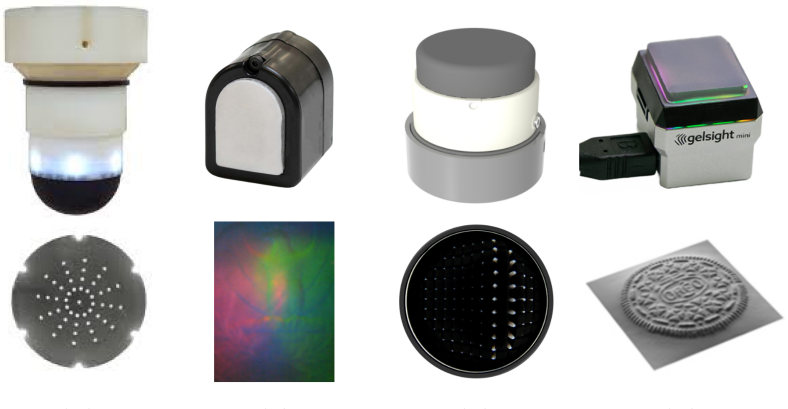
Chapter 1 introduced the central question of this dissertation: whether short histories of contact information can provide an early warning of near-future grasp instability. To place that question on firmer ground, it is first necessary to review the relevant literature. The work most closely related to this dissertation spans tactile sensing, contact mechanics and slip, reactive grasp monitoring, predictive grasp assessment, and simulation-based learning. This chapter reviews those areas in order to show what is already established, where the main limitations remain, and why predictive grasp-stability forecasting remains a distinct problem.

2.1 TOUCH AS A SOURCE OF GRASP INFORMATION

During robot manipulation, vision and touch provide complementary information. Vision supplies global scene information, such as object pose, scene layout, and free space, whereas touch provides local information at the contact area. Recent benchmarks confirm that no single modality works best in every setting. Luu et al. [14] show that tactile input becomes especially useful when visual information becomes ambiguous or occluded, whilst Liu et al. [15] show that even sparse tactile signals improve dexterous manipulation performance when combined with vision.

This distinction matters for grasp stability. Vision can guide the hand towards an object, but once contact begins, the success of the grasp depends on what happens at the point of contact. From that point, touch reveals the contact location, changes in contact force, and any motion of the object relative to the hand. Touch does more than supplement vision. It directly indicates whether the grasp remains secure.

Vision-based tactile sensors provide richer contact information than older sensors measuring only force and torque. Sensors such as GelSight [6], DIGIT [7], TacTip [8], and FingerVision [9] yield dense observations of contact deformation, marker motion, and local geometry. Li et al. [16] show that these sensors can estimate shape, contact geometry, and force-related properties. NeuralFeels [17] employs visuo-tactile input to estimate object shape and pose during in-hand manipulation, TouchSDF [18] reconstructs 3D geometry from tactile observations, and AnyRotate [19] utilises dense tactile sensing to recover from unstable grasps while rotating unseen objects. Collectively, these studies show that tactile sensing captures information directly relevant to secure object manipulation.



(A) (B) (C) (D)

Figure 2.1: Examples of tactile sensors and their tactile reading representations. (A) TacTip [8], (B) DIGIT [7], (C) BioTacTip[20], (D) GelSight [6].

2.2 CONTACT MECHANICS, SLIP, AND GRASP STABILITY

When a fingertip presses on an object, the contact force comprises two components: a normal force acting perpendicular to the surface and a tangential or shear force acting along it. Under Coulomb friction, the shear force must remain within the friction cone defined by the coefficient of friction and the available normal force [21]. If the shear force exceeds the limit imposed by the available normal force, the contact starts to slide.

This clarifies why grip force alone fails to guarantee a stable grasp. Taylor et al. [22] note that slip occurs when grasping force and friction prove insufficient to maintain force closure, preventing the grasp from resisting external disturbance. In practice, though, force levels represent only one component of the problem. Contact location, contact area, object shape, and the moments produced about the object also dictate grasp robustness; thus, a grasp might apply a large force yet remain unstable if the contacts occupy poor positions.

Chen et al. [12] distinguish between incipient slip and gross slip. In incipient slip, some parts of the contact area begin to move while other parts remain fixed. In gross slip, the object slides relative to the sensor. This difference matters because gross slip indicates obvious failure, whereas incipient slip acts as an earlier warning sign.

These ideas show that grasp stability depends on more than grip force alone. It depends on how normal force, shear force, friction, and contact location interact during contact.

2.3 REACTIVE MONITORING AND PREDICTIVE GRASP ASSESSMENT

Much of the tactile grasping literature frames stability as a reactive monitoring problem. In this setting, the robot monitors tactile signals during or after contact, detects slip once it begins, and then adjusts grip force or hand pose to recover. For vision-based tactile sensors, this often involves tracking motion on the sensor surface. Dong et al. [23] used GelSight to detect translational and rotational slip from relative motion, marker displacement, and changes in contact area. Dong et al. [24] later monitored incipient slip with GelSlim by comparing the observed deformation field with the motion expected from a rigid contact patch, enabling earlier warning before gross slip. Lu et al. [13] similarly classify no-slip, incipient slip, and gross slip states with a NeuroTac sensor. These methods provide valuable warning signals, but they still operate after local slip has already started.

Reactive tactile sensing also supports closed-loop grasp control. Ford et al. [25] use tactile estimates of contact pose and force to regulate grip force continuously in a multi-fingered hand. Jawale et al. [26] combine slip detection with slip-severity estimation to guide corrective control without overcompensating. Ayral et al. [27] combine fast slip cues with contact localisation to update internal forces online in a multifingered grasp. AnyRotate [19] shows that dense tactile sensing can help a policy detect unstable grasps and recover while manipulating unseen objects. Although these systems can preserve or recover a grasp, they remain reactive, because their control logic begins once instability has already become visible in the tactile signal.

Predictive work asks a different question: can a system judge grasp security before failure becomes obvious? Calandra et al. [28] show that post-contact visuo-tactile observations improve grasp-outcome prediction relative to vision alone, demonstrating that touch helps assess a grasp's chance of success. Zapata-Impata et al. [29] provide a more direct example of predictive grasp assessment, using BioTac measurements to predict grasp stability and slip before lift. These studies mark important steps towards anticipatory grasping, but they also expose limitations relevant to the present dissertation: the work relies on strongly sensor-specific observations, the prediction targets remain coarse, and collecting large, well-labelled predictive datasets on hardware remains slow and labour-intensive.

This literature separates three related tasks: predicting grasp outcome after contact, detecting incipient slip once local motion has begun, and controlling grasps reactively from tactile feedback. The narrower forecasting problem addressed here has received less attention: whether short histories of normal and shear force can predict that an ongoing grasp will become unstable soon, before explicit slip cues appear.

2.4 SIMULATION–BASED LEARNING AND GRASP GENERATION

Training and testing grasping systems directly on hardware proves slow, labour-intensive, and difficult to scale. Simulation has thus emerged as an essential tool in robot learning, particularly for trials requiring high-volume data under controlled conditions. General-purpose simulators such as MuJoCo [30] and Isaac Lab [31] support this broader trend, while TACTO [32] and Tactile Gym 2.0 [33] focus more directly on tactile sensing. For grasp-stability prediction, simulation facilitates the generation of many candidate grasps, enables their controlled replaying, and provides the contact data and outcome labels required for learning and evaluation.

TACTO is especially relevant here because it demonstrates that simulated tactile sensing can support large-scale predictive learning. Wang et al. [32] used the simulator to train a grasp-stability predictor from one million simulated grasps, illustrating how simulation can scale tactile dataset generation far beyond what is practical with physical hardware alone.

Simulation also supports policies that transfer to real hardware. Su et al. [34] show that tactile policies trained in simulation generalise across unseen objects to real robots. Yet, transfer remains non-trivial. Luo et al. [35] argue that tactile sensing resists accurate simulation due to reliance on deformable media, contact mechanics, and the complex process transforming deformation into electrical or image-like signals. Friction, compliance, contact force, and shear complicate exact matching, creating a sim-to-real gap. Researchers must view simulation as a practical method for generating controlled predictive datasets, rather than proof that models trained in simulation function unchanged on physical hardware.

Building a stability-prediction model also necessitates a method for generating plausible grasps for data collection. Here, the relevant literature focuses on dexterous grasp synthesis. Liu et al. [36] incorporate force closure directly into differentiable grasp optimisation, enabling the refinement of hand poses towards grasps that remain physically stable and geometrically sound. DexGraspNet [37] scales this approach into a dataset-generation pipeline by combining force-closure-inspired objectives with constraints on contact distance, penetration, self-collision, and joint limits.

This literature matters because the data used here should come from plausible candidate grasps rather than arbitrary hand-object collisions. By generating grasps under constraints on contact, friction, collision, and feasibility, the later prediction task can focus on when a plausible grasp becomes unstable.

2.5 RESEARCH GAP AND SCOPE

The literature points to two gaps relevant to this dissertation. First, much of the work on grasp stability either detects slip after it has started or predicts coarse grasp outcomes from sensor-specific observations. Fewer studies ask whether an apparently stable grasp will become unstable within a short future interval while no explicit slip cue is yet visible. Secondly, predictive tactile methods often depend on raw tactile images, deformation fields, or feature encodings tied closely to a particular sensor. The literature still does not show clearly whether simpler contact signals, such as normal and shear force, are enough for this forecasting task.

This dissertation addresses a narrower question: can short temporal windows of normal and shear contact forces provide an early warning that a grasp will become unstable? It studies that question in simulation through a pipeline that combines dexterous grasp generation, contact-feature extraction, temporal labelling, and evaluation across a range of objects, including objects not seen during training.

3 METHODS

Chapter 2 identified the main gap addressed in this dissertation: whether short temporal windows of sensor-invariant contact information can provide an early warning of grasp instability. Answering that question requires more than a prediction model alone. It also requires a method for generating plausible grasps, replaying them under controlled conditions, extracting meaningful contact features, and assigning labels that refer to future instability rather than only to the present state. This chapter therefore describes the simulation-based pipeline used to turn that research question into a supervised prediction problem.

3.1 GRASP GENERATION

The grasp-generation stage draws on DexGraspNet [37]. It adapts that framework for the ORCA hand [38] instead of the ShadowHand [39], MANO [40], and Allegro [41]. It also simplifies some operations so the pipeline runs reliably on laptop-class hardware. The method keeps the main idea of energy-based grasp synthesis while replacing expensive components with lighter alternatives.

At a high level, the grasp-generation pipeline has five parts. First, the pipeline represents the object and hand in a form that suits MuJoCo simulation and geometric optimisation. Secondly, it defines a discrete set of plausible contact locations on the hand, so the energy focuses on mechanically meaningful regions. Thirdly, a structured pre-grasp places the open hand near the object with a sensible approach direction. Fourthly, the optimiser refines this pre-grasp by minimising an energy function to produce a physically plausible grasp candidate. Finally, the pipeline replays the resulting candidates in simulation and filters them so that only physically usable grasps move to the data-generation stage. The following subsections describe these parts in order.

3.1.1 Object and Hand Representation

(a) a_toy_airplane **(b)** cracker_box **(c)** large_marker

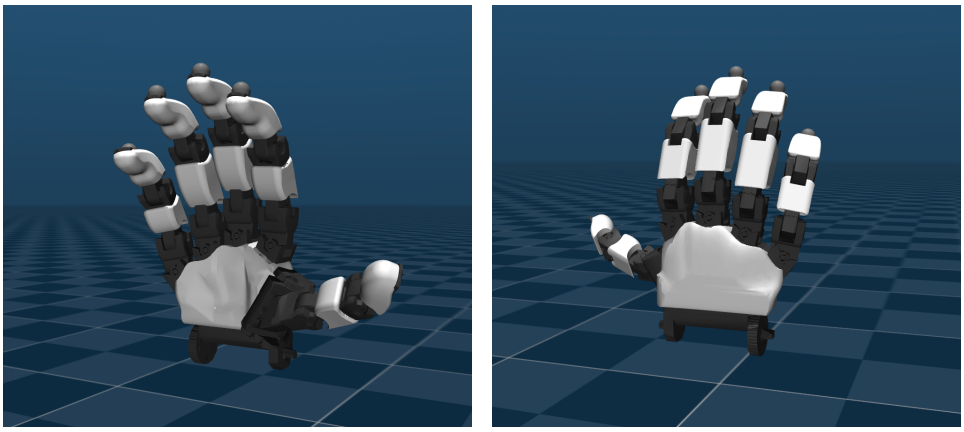
Figure 3.1: Sample of three YCB Objects in MuJoCo after pre-processing.

Grasp generation needs digital models of both the object and the hand. These models give the simulator a three-dimensional description of each item's shape and physical properties, such as mass and size. Researchers have developed many object datasets for robotic manipulation, including YCB [42], GraspFactory [43], and Google Scanned Objects [44]. These datasets commonly provide 3D meshes and textures for a range of objects, either from scans of real items or from synthetic modelling. This work uses the YCB object dataset for two reasons. First, robotic grasping researchers

use it widely, which makes comparison with earlier studies easier. Secondly, many YCB objects also have physical counterparts, which supports possible future sim-to-real transfer.

This work uses MuJoCo, but the YCB meshes need pre-processing before they can run in the simulator. The pre-processing pipeline first repairs surface issues such as holes, inverted faces, and incorrect normal directions. It then places the object in a consistent upright reference frame and re-centres it at its centre of mass. Finally, it assigns object mass and diagonal inertia, and splits the mesh into a small number of convex parts that act as collision meshes in MuJoCo. Figure 3.1 illustrates three random YCB objects within MuJoCo.

Grasp generation and simulation use the ORCA hand model. ORCA has a multi-fingered robotic design with an anthropomorphic shape. It combines fingertip tactile sensing with finger geometry that stays close to the finger's original form. This combination differs from many vision-based tactile systems, where the sensing hardware can grow bulky and can change the fingertip shape. That combination makes ORCA a good platform for studying grasp stability from contact information in a hand form closer to a natural finger. The ORCA hand description repository provides MuJoCo-compatible hand-description files[1], so the pipeline needs no extra hand-model pre-processing. Figure 3.2 shows the ORCA hand in MuJoCo using the description files, modified to not include the base/wrist.



(a) Front **(b)** Back

Figure 3.2: ORCA hand model in MuJoCo, modified to exclude the wrist/base.

The ORCA hand has 17 degrees of freedom (DoF): 16 finger-joint DoF plus the wrist. During grasp generation, the wrist is treated as a free pose, while a 16-dimensional joint vector describes the finger joint angles,

$$\theta \in \mathbb{R}^{16}, \tag{1}$$

A grasp configuration, g, combines the wrist pose and the finger joint angles:

$$g = (P, R, \theta), \tag{2}$$

where $P \in \mathbb{R}^3$ denotes the wrist position, $R \in \mathbb{R}^6$ gives a continuous 6-D parameterisation of wrist orientation, and θ specifies the finger joint angles. The 6-D orientation parameterisation gives a smooth, unconstrained representation for grasp generation. It avoids the discontinuities of Euler angles and the unit-norm constraint of quaternions. Internally, the system converts this representation into a rotation matrix and then a quaternion for use in MuJoCo.

3.1.2 Contact Candidate Set

Grasp synthesis needs a finite set of contact candidate locations on the hand. Evaluating contact over the whole hand mesh would require too much computation and would let the energy function focus on regions unlikely to matter in a grasp. This method searches only a discrete set of plausible contact locations on the ORCA hand, following the same general idea as DexGraspNet.

1 https://github.com/orcahand/orcahand_description

When grasp generation starts, the method builds these candidate locations from the ORCA collision meshes. It uses ten contact-bearing regions in total, with two on each finger. Within each region, it identifies a palm-facing patch and selects 12 candidate contact points using farthest-point sampling. This gives a fixed set of $N = 120$ candidate contact points for each optimisation run. The sampling strategy spreads the points across the available contact area instead of letting them cluster in one small part of the surface. After this step, the candidate set stays fixed for that optimisation run. Figure 3.3 shows a sample of these points on the ORCA hand.

The contact candidates serve two purposes. First, they mark the hand locations where the method computes grasp energy. Secondly, they keep the optimisation focused on surfaces that can realistically make contact during a grasp. This encourages physically plausible grasp configurations instead of letting arbitrary or mechanically irrelevant parts of the hand geometry dominate the energy.

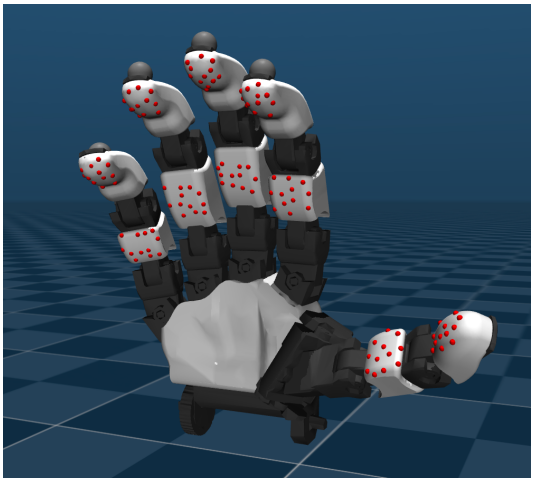


Figure 3.3: Contact candidate points (red spheres) on the ORCA hand, selected by farthest-point sampling across the palm-sided surfaces of the 10 available regions.

3.1.3 Pre-grasp Initialisation

The optimiser does not start from an arbitrary hand pose. A purely random initialisation would waste many iterations on poses where the palm faces away from the object, the fingers start too closed, or the hand already sits in self-collision. The pipeline instead samples a structured pre-grasp that places an open ORCA hand near a plausible approach direction before optimisation begins.

Let O denote the transformed object mesh and let c denote its centroid. The sampler first computes the convex hull of O and inflates it by moving each hull vertex away from c by a fixed distance ρ. It then samples a point a on this inflated hull. The nearest point s on the original object surface defines an inward surface direction:

$$n = \frac{s - a}{\|s - a\|_2}. \tag{3}$$

The sampler then jitters this direction within a cone of half-angle α to get a candidate approach direction $\hat{n}$. It orients the wrist so that the palm faces along $\hat{n}$, applies a random roll angle about that axis, and translates the wrist backwards from the candidate contact region by a random standoff distance d. In the implementation, it places the hand so that a fixed grasp-focus point in the palm lies near the sampled anchor. This gives the initial pose a clear geometric meaning: the fingertips point towards a plausible contact region, while enough free space remains for the optimiser to refine the grasp.

The current implementation uses an inflation distance of 0.08 m, a standoff range of 0.06 m to 0.10 m, and a cone half-angle of 30°. The sampler rejects candidate approach directions with a positive vertical component, so it does not favour unrealistic upward approaches from underneath the object. Unlike DexGraspNet, the method does not jitter the finger joints around a reference posture. Instead, the hand starts from a fixed open configuration chosen to keep the grasp cavity available for the later optimisation stage.

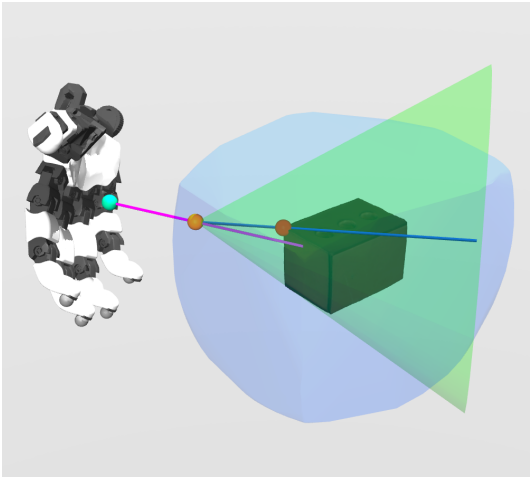


Figure 3.4: Example of a pre-grasp initialisation for the foam brick object. The blue hull shows the inflated object convex hull. An orange anchor point marks a sample on that hull, and the nearest point on the true object surface (red) defines an inward approach direction (magenta). The method then places the hand with a random roll and standoff while the fingers remain in a fixed open posture (teal).

3.1.4 Energy-based Optimisation

From a sampled pre-grasp, g_0, the optimiser searches for a configuration with low energy. Low energy means that contact candidate regions lie close to the object, the resulting contacts oppose forces effectively, collisions stay small, and joint angles remain within limits. The total energy takes the form

$$E(g) = w_{dis}E_{dis} + w_{fc}E_{fc} + w_{pen}E_{pen} + w_{spen}E_{spen} + w_{joint}E_{joint}. \tag{4}$$

The energy has the following terms.

DISTANCE ENERGY. At each optimisation step, the current hand pose transforms the $N = 120$ contact candidate points, denoted $p_1, p_2, \ldots, p_N$ with $p_i \in \mathbb{R}^3$, from the hand frame into world coordinates. For each transformed point, the signed distance $d(p_i)$ measures the distance to the object surface. The value equals zero on the surface, takes positive values outside the object, and takes negative values inside it. The distance energy becomes

$$E_{dis} = \sum_{i=1}^{N} |d(p_i)|. \tag{5}$$

This term reaches its smallest value when a contact candidate lies on, or close to, the object surface. Taking the absolute value penalises both a gap between the hand and object and penetration into the object. This encourages the optimiser to move the candidate contact regions towards gentle surface contact instead of leaving them far away or pushing them deeply into the object.

FORCE-CLOSURE ENERGY. The signed-distance calculation also provides a local surface normal for each transformed contact candidate. Reversing this direction gives an inward-pointing contact normal, $c_i \in \mathbb{R}^3$. In the implementation, this inward normal gives the direction of the force that contact would apply to the object. Stacking these directions gives the vector

$$c = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_N \end{bmatrix} \in \mathbb{R}^{3N}. \tag{6}$$

To check how well the full set of candidate contacts can balance the object, the method uses the grasp matrix

$$G = \begin{bmatrix} I_3 & \cdots & I_3 \\ [p_1]_\times & \cdots & [p_N]_\times \end{bmatrix}, \tag{7}$$

where p_i denotes the position of the ith candidate contact point, and $[p_i]_\times$ denotes the matrix form of the cross product with p_i:

$$p_i = \begin{bmatrix} p_{i,x} \\ p_{i,y} \\ p_{i,z} \end{bmatrix}, \qquad [p_i]_\times = \begin{bmatrix} 0 & -p_{i,z} & p_{i,y} \\ p_{i,z} & 0 & -p_{i,x} \\ -p_{i,y} & p_{i,x} & 0 \end{bmatrix}. \tag{8}$$

In the grasp matrix, I_3 denotes the 3×3 identity matrix. For each candidate point p_i, this matrix maps a contact-force direction applied at that point to its contribution to the net force and net moment about the object. The repeated identity blocks in the top row of G pass each contact vector through unchanged, so the block-matrix multiplication sums them to give the resultant force. The bottom row uses $[p_i]_\times$ to convert each contact vector into the corresponding moment, $p_i \times c_i$, and then sums those moments.

Applying the grasp matrix in eq. (7) to the stacked contact-force directions in eq. (6) gives the combined force and moment generated by all candidate contacts:

$$Gc = \begin{bmatrix} I_3 & \cdots & I_3 \\ [p_1]_\times & \cdots & [p_N]_\times \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_N \end{bmatrix} = \begin{bmatrix} F_x \\ F_y \\ F_z \\ M_x \\ M_y \\ M_z \end{bmatrix}. \tag{9}$$

$$E_{fc} = F_x^2 + F_y^2 + F_z^2 + M_x^2 + M_y^2 + M_z^2. \tag{10}$$

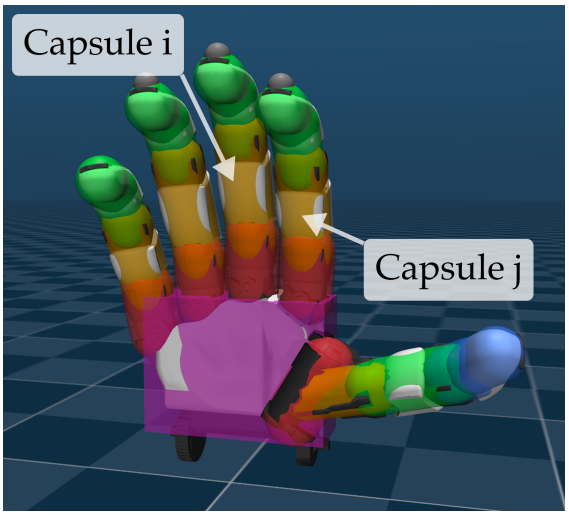
Here, F_x, F_y, and F_z denote the components of the resulting net force, while M_x, M_y, and M_z represent the components of the resulting net moment. Under this approximation, the force-closure energy in eq. (10) stays small when the candidate contacts roughly balance one another, leaving little residual force or moment. A low value shows good force and moment balance, but it does not guarantee grasp stability.

PENETRATION ENERGY. The distance energy encourages candidate contact points to move towards the object surface, but it does not stop other parts of the hand from passing through the object. A separate penetration term penalises overlap between the object and the rest of the hand.

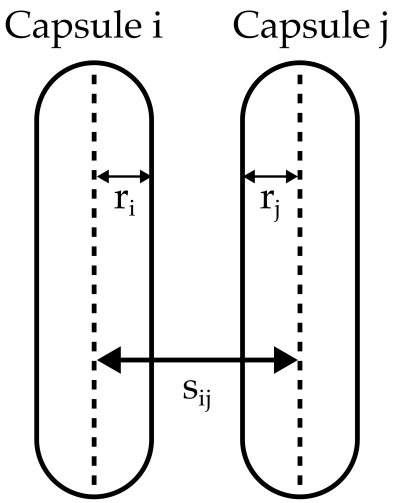
To measure this, the method follows the reverse-penetration idea used in DexGraspNet. Instead of sampling points on the hand and asking whether they lie inside the object, it samples points on the object surface and asks whether they lie inside the hand. This approach handles thin or imperfect object meshes more robustly. Here, the method replaces the object with its convex hull and samples 2000 surface points from it. If S_o denotes the set of sampled points on the object surface and $d_H(q)$ denotes the signed distance from q to the hand, then

$$E_{pen} = \sum_{q \in S_o} \max(-d_H(q), 0). \tag{11}$$

A sampled point contributes zero when it lies outside the hand or exactly on its surface. It contributes a positive penalty only when it lies inside the hand, in which case the penalty equals its penetration depth. This term stays small when the sampled object-surface points remain outside the hand and increases as the object becomes embedded in the hand. To keep the computation lightweight, the method approximates the hand signed-distance field analytically using capsules for the finger links and a box for the palm, rather than an expensive full mesh query.



(a) Capsules and a box representing the simplified geometry of the ORCA hand's fingers and palm.



(b) Side-view diagram showing the capsule centre segments (dashed lines), capsule radius r_j, and separation s_{ij}.

Figure 3.5: Simplified geometry used in the penetration and self-penetration energy terms.

SELF-PENETRATION ENERGY. The optimiser must also avoid configurations where different fingers pass through one another. Using the capsule approximation shown in fig. 3.5a, the method computes the shortest separation between capsule centre segments, s_{ij}, for each valid pair, as illustrated in fig. 3.5b. If capsules i and j have radii r_i and r_j, then their surface-to-surface gap equals

$$g_{ij} = s_{ij} - r_i - r_j, \tag{12}$$

and the self-penetration energy becomes

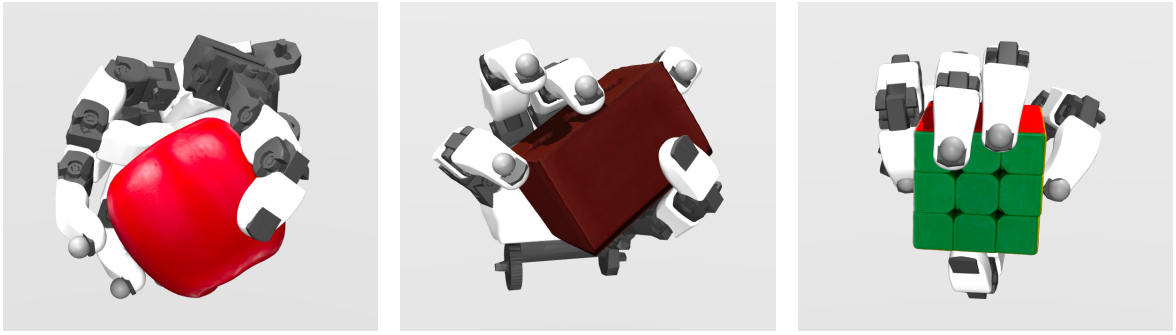
$$E_{spen} = \sum_{(i,j)\in\mathcal{P}} \max(-g_{ij}, 0)\,. \tag{13}$$

Here, capsules i and j denote two simplified hand links. The set $\mathcal{P}$ lists the capsule pairs checked for collision. It compares links from different fingers and skips mechanically connected links on the same finger. The quantity s_{ij} gives the shortest straight-line separation between the centre segments of capsules i and j. The gap g_{ij} takes positive values when the capsules lie apart, zero when they just touch, and negative values when the capsules penetrate one another. The term $\max(-g_{ij}, 0)$ contributes zero when no collision occurs and a positive penalty when one capsule penetrates another. This term stays small when different fingers remain clear of one another throughout the grasp.

JOINT-LIMIT ENERGY. The optimiser should also keep each finger joint within its allowed range. Let θ_k denote the current angle of joint k, with lower and upper limits θ_k^{min} and θ_k^{max}. The joint-limit energy becomes

$$E_{joint} = \sum_{k=1}^{16} \left[\max(\theta_k - \theta_k^{max}, 0) + \max\left(\theta_k^{min} - \theta_k, 0\right)\right]. \tag{14}$$

Each term contributes zero when the joint lies within its allowed range and increases linearly with the amount of violation once the joint moves beyond either limit. These lower and upper bounds come from the ORCA hand description model files used to build the simulator. The implementation also clips every candidate proposal to the allowed joint range before evaluation. This keeps the search stable and avoids wasting iterations on impossible finger postures.



(a) Apple **(b)** Foam Brick **(c)** Rubik's Cube

Figure 3.6: Examples of randomly selected grasps developed from the grasp generation pipeline.

PRACTICAL DEVIATIONS FROM DEXGRASPNET. This implementation differs from DexGraspNet in two main ways. First, the full bank of 120 candidate contact points stays fixed throughout each optimisation run, rather than augmenting the state with a smaller set of explicitly resampled contacts. Secondly, the optimiser itself uses a simpler design. DexGraspNet uses a differentiable formulation with Langevin-style updates, whereas this project uses simulated annealing.

Starting from the sampled pre-grasp g_0, each optimisation step applies an independent Gaussian update to the wrist position, wrist orientation, and finger joint coordinates. The implementation uses standard deviations of 0.005 m for wrist position, 0.02 for the 6-D wrist orientation representation, and 0.05 rad for the joint angles. After each proposed update, the implementation clips the finger joints back into their allowed ranges before evaluating the energy.

If a proposal lowers the total energy, the optimiser accepts it immediately. Otherwise, it may still accept the proposal with Boltzmann probability

$$P(\text{accept}) = \exp\left(-\frac{\Delta E}{T}\right), \tag{15}$$

where ΔE denotes the increase in energy and T denotes the current temperature. This means the optimiser can still accept a higher-energy proposal, but the chance drops as the energy increase grows or the temperature falls. In the implementation, T decays exponentially from 1000 to 1 over 6000 iterations. The optimiser keeps track of the best grasp found anywhere along the search trajectory, rather than simply returning the final state. The system then executes parallel optimisation runs across CPU processes so that grasp generation remains practical on laptop hardware. The energy weights used here follow

$$(w_{\text{dis}}, w_{\text{fc}}, w_{\text{pen}}, w_{\text{spen}}, w_{\text{joint}}) = (100, 1, 10, 100, 1000). \tag{16}$$

These values do not match the default DexGraspNet weights, which effectively use $(100, 1, 100, 10, 1)$ in the same order. Here, the weights place much stronger emphasis on joint feasibility and self-collision avoidance, and less emphasis on object penetration. In practice, this helps the optimiser reject implausible ORCA-hand grasps quickly while still encouraging the candidate contact regions to approach the object surface and balance forces reasonably well.

3.2 GRASP REPLAY AND DATA RECORDING

The grasp-generation stage returns static hand configurations, but the prediction models receive time-varying contact signals as inputs. The next stage of the present pipeline therefore replays each candidate grasp in MuJoCo with physics enabled. Each replay starts from an already formed grasp and then applies gravity to test whether the grasp stays stable. If it does, the contact signals should remain steady. If the object starts to move or slip, those signals should change over time.

For replay, the system loads the ORCA hand directly into the optimised grasp pose. It sets the finger joints to the optimised joint angles and holds those same joint targets throughout the simulation. The replay uses no separate reaching, closing, or lifting controller. This means the replay tests the quality of the grasp itself rather than the behaviour of a control policy.

At the start of replay, the system sets gravity to zero while it places the hand and object in the optimised configuration. The system then switches gravity to

$$g = (0, 0, -9.81)\,\mathrm{m/s^2}, \tag{17}$$

so that only downward gravity acts during replay. This replay condition differs from DexGraspNet, which validates grasps across six gravity directions. Each replay runs for 40000 time steps. With a MuJoCo time step of $dt = 5 \times 10^{-4}$ s, this corresponds to about 20 seconds of simulated time per grasp.

At every time step, the replay records the object position p_t, the object linear velocity v_t, and the smallest signed distance between the hand and object, d_t. The recorded quantity d_t does not give a single distance for the whole hand. More precisely, d_t gives the smallest signed distance among the object–hand contact pairs that MuJoCo is currently tracking. If MuJoCo reports no object–hand contacts at time t, the method marks d_t as undefined. Finite values of d_t show that some hand–object contact exists, while missing values mark contact loss. The replay also records fingertip force features for the thumb, index, middle, ring, and little fingers.

The model uses only contacts on the five fingertip bodies as inputs. This reflects the main sensing assumption in this work: the predictive model uses fingertip tactile information rather than a full contact map over the whole hand. Contacts on the palm or other finger links still affect the replay physics, but the model does not include them in the learning features.

Let $C_{j,t}$ denote the set of object–hand contacts assigned to fingertip j at time t. MuJoCo breaks each contact force $k \in C_{j,t}$ into one normal component, $N_{k,t}$, and two tangential components, $A_{k,t}$ and $B_{k,t}$. The method then records fingertip normal and shear forces as

$$F_{\mathrm{normal},j,t} = \sum_{k \in C_{j,t}} \left| N_{k,t} \right|, \qquad F_{\mathrm{shear},j,t} = \sum_{k \in C_{j,t}} \sqrt{A_{k,t}^2 + B_{k,t}^2}. \tag{18}$$

Here, $N_{k,t}$ acts along the local contact normal, while $A_{k,t}$ and $B_{k,t}$ lie in the local contact plane. The method sums all object contacts on the same fingertip to give one normal-force signal and one shear-force signal for that finger.

This aggregation is a modelling choice rather than a direct simulation of raw tactile images. A real vision-based tactile sensor often provides spatially distributed information across the fingertip surface. Here, the method reduces that detail to fingertip-level normal and shear forces, so the predictive models use compact contact features rather than raw tactile images or simulator-specific contact pairs.

The feature vector at time t is

$$x_t = \left[F_{\mathrm{normal},1,t}, F_{\mathrm{shear},1,t}, \ldots, F_{\mathrm{normal},5,t}, F_{\mathrm{shear},5,t} \right] \in \mathbb{R}^{10}. \tag{19}$$

Each replay becomes a sequence of ten force signals. The prediction models use this force sequence as input. The method keeps the object-motion and signed-distance traces separately and uses them in the next section to define the stable and unstable windows for supervised training.

3.3 WINDOW LABELLING

A replayed grasp rarely stays in one condition for the full simulation. Some replays remain quiet from start to finish, some become unstable almost immediately, and many begin stably before later degrading. Assigning one label to the entire replay would discard the timing information needed for prediction. Instead, the method handles the replay in three stages: first, it identifies the usable part of the trajectory; secondly, it marks each timestep in that interval as stable or unstable; and thirdly, it samples short force windows and gives them predictive labels.

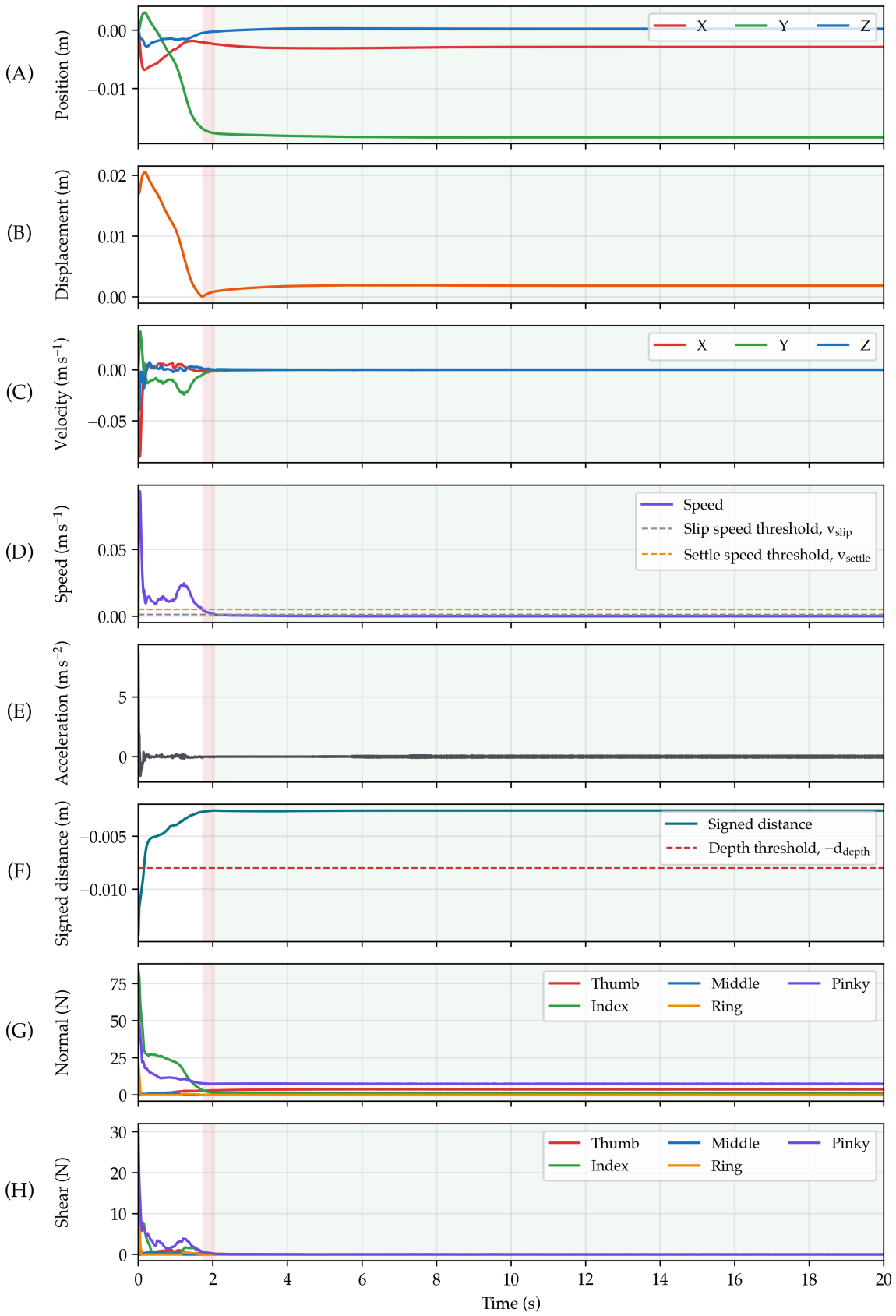


Figure 3.7: Replay data for the apple grasp shown in fig. 3.6a. Panels show (A) object position, (B) displacement from the post-settling reference position, (C) object velocity v_t, (D) object speed $\|v_t\|_2$, (E) the time derivative of object speed, (F) minimum hand–object signed distance d_t, (G) aggregated fingertip normal force, and (H) aggregated fingertip shear force. For each fingertip, normal force is the summed absolute contact-normal force, while shear force is the summed two-dimensional tangential magnitude in the local contact plane, as defined in eq. (18). The signed-distance settling condition is given in eq. (22). Shaded regions show unstable periods (red) and stable periods (green) based on the instability definition in eq. (23).

3.3.1 Usable Replay Interval

Not every timestep is suitable for training. Immediately after gravity turns on, the object may shift slightly before the grasp settles. Windows drawn from this initial transient mainly reflect replay initialisation rather than grasp quality. The usable interval begins only after a short settling stage and ends when contact ends.

The method uses two settling checks within the first 3 seconds of the replay. The first check uses speed and searches for the earliest interval of duration 0.1 seconds during which

$$\|v_t\|_2 \leqslant v_{\text{settle}}, \qquad v_{\text{settle}} = 5 \text{ mm/s}, \tag{20}$$

so the object must remain slow for a sustained period rather than for a single isolated timestep.

The second check uses the hand–object signed distance and looks for the earliest interval of duration 0.1 seconds during which the contact geometry changes little. To estimate the settled distance, let $\tilde{d}$ denote the median of the finite signed-distance values in the final 0.2 seconds of the 3 second search interval, and let

$$\tau_d = \max(0.5 \text{ mm},\ 3 \times 1.4826 \times \text{MAD}), \tag{21}$$

where MAD is the median absolute deviation about $\tilde{d}$. Here, τ_d is the tolerance threshold around the reference distance $\tilde{d}$. The term $1.4826 \times \text{MAD}$ gives a robust estimate of standard deviation, and the factor of 3 defines a three-sigma tolerance band [45]. The signed-distance condition is then

$$|d_t - \tilde{d}| \leqslant \tau_d. \tag{22}$$

If both checks succeed, the method uses the later of the two indices as the settle point. This means windows do not begin until both object motion and contact geometry have stabilised. If neither check succeeds, the implementation falls back to the 3 second boundary as a conservative warm-up point.

After this warm-up point, the method filters the replay in two further ways. First, it rejects a replay if the signed distance remains below the depth threshold $-d_{\text{depth}}$, where $d_{\text{depth}} = 8$ mm, for at least 1 second. In practice, such trajectories reflect unrealistic overlap between the hand and object rather than a meaningful grasp. Secondly, the usable interval ends at the first sustained loss of contact. If the signed distance remains undefined for 0.1 seconds after settling, the method takes the first timestimestep-wise ruleep of that missing-contact run as the contact end.

3.3.2 Timestep-wise Stability Labels

After the settle point and before contact ends, the method treats the remaining part of the replay as the usable interval. Within this interval, each timestep receives an instability indicator based on object speed:

$$u_t = \mathbb{I}\left[\|v_t\|_2 \geqslant v_{\text{slip}}\right], \qquad v_{\text{slip}} = 1.2 \text{ mm/s}, \tag{23}$$

where $\mathbb{I}[\cdot]$ is the indicator function. Here, $u_t = 1$ marks the replay as unstable at timestep t, while $u_t = 0$ marks that timestep as stable. The threshold $v_{\text{slip}} = 1.2$ mm/s is used here as a fixed labelling rule rather than as a tuned result-specific parameter. In the implementation, the same speed threshold is applied unchanged across all objects, model families, and unseen-object splits. This keeps the predictive task definition fixed, so later performance differences can be interpreted as differences in model behaviour rather than as changes in how instability itself is labelled.

These timestep-wise labels do not yet define model samples. They simply describe the state of the replay at each timestep. Consecutive timesteps with $u_t = 1$ form unstable periods, and the remaining periods form stable ones. The next stage builds predictive labels from these timestep-wise labels. Figure 3.7 shows these periods as shaded regions.

3.3.3 Window Construction

The method then samples model inputs from the same usable interval. Each sample uses three time spans: an observation window, a predictive offset, and a future horizon used to assign the label. Let

τ_w, τ_o, and τ_h denote their durations respectively. With simulation step size Δt, the corresponding step counts become

$$W = \left\lceil \frac{\tau_w}{\Delta t} \right\rceil, \qquad O = \left\lfloor \frac{\tau_o}{\Delta t} + \frac{1}{2} \right\rfloor, \qquad H = \left\lceil \frac{\tau_h}{\Delta t} \right\rceil. \tag{24}$$

In the baseline configuration,

$$\tau_w = 0.2 \text{ s}, \qquad \tau_o = 0.1 \text{ s}, \qquad \tau_h = 0.15 \text{ s}, \tag{25}$$

which gives $W = 400$, $O = 200$, and $H = 300$ time steps at the current simulator step size. These values are best understood as fixed benchmark-design choices. Together, they define a short-history forecasting task: the observation window gives the model a brief recent force history, the predictive offset requires a genuine lead time before the target interval begins, and the short future horizon keeps the label local to the near future rather than to the full replay. The implementation keeps τ_o and τ_h fixed across the later comparisons so that changes in performance can be attributed to model choice, unseen objects, or observation-window length rather than to a moving task definition.

For every valid grasp, the method chooses candidate window starts from

$$t_{\text{settle}} \leqslant s \leqslant t_{\text{contact\_end}} - W - O - H. \tag{26}$$

Here, s denotes the start index of one sampled observation window. By default, the method samples up to 20 distinct start indices from each valid grasp, with no start point chosen more than once. When possible, it always includes the window that begins exactly at the settle point, and then samples the remaining windows later in the same replay. This lets one replay provide many supervised examples while still keeping all windows from that replay in the same dataset split. The cap of 20 sampled windows per grasp is also fixed across the reported experiments. This allows one replay to contribute more than one supervised example while preventing a small number of long replays from contributing an arbitrarily large share of the full dataset.

For a chosen start index s, the input tensor is

$$X_s = [f_s, f_{s+1}, \ldots, f_{s+W-1}], \tag{27}$$

where X_s is the full observation window beginning at timestep s, and $f_t \in \mathbb{R}^{5\times C}$ is the fingertip-feature matrix at timestep t. The quantity C denotes the number of feature channels per fingertip. In the default setting, $C = 2$, using the normal and shear forces defined in eq. (18), so

$$f_t = \begin{bmatrix} F_{\text{normal},1,t} & F_{\text{shear},1,t} \\ \vdots & \vdots \\ F_{\text{normal},5,t} & F_{\text{shear},5,t} \end{bmatrix}. \tag{28}$$

3.3.4 Predictive Labels

The final label depends on what happens after the observation window, not on the current state inside it. In other words, the label answers the question: if the model observes the force history in X_s, does the grasp become unstable shortly afterwards?

For a window beginning at timestep s, the future target interval begins after both the observation window and the predictive offset, so its first timestep is $s + W + O$. Let T_s denote the set of future timesteps from that point up to whichever comes first: the end of the prediction horizon or the end of contact. If no future timesteps remain, the method discards the sample.

Each timestep $k \in T_s$ already has an instability indicator u_k defined by eq. (23). The proportion of the target interval that is unstable is then

$$r_s = \frac{1}{|T_s|} \sum_{k \in T_s} u_k, \tag{29}$$

where $r_s = 0$ means the whole target interval is stable and $r_s = 1$ means the whole target interval is unstable.

The symbol y_s denotes the final predictive label for the window that starts at s, where $y_s = 0$ denotes a stable future interval and $y_s = 1$ denotes an unstable future interval. Using $\gamma_{low} = 0.35$ and $\gamma_{high} = 0.65$, the method assigns the label as

$$y_s = \begin{cases} 1, & r_s \geqslant \gamma_{high}, \\ 0, & r_s \leqslant \gamma_{low}, \\ \text{discarded}, & \gamma_{low} < r_s < \gamma_{high}. \end{cases} \tag{30}$$

This ambiguity band excludes windows containing a mixture of stable and unstable behaviour. The label depends on the future horizon rather than the current state within the observation window. Figure 3.8 shows the opposite case: the horizon contains both stable and unstable periods, but only 28.24% of that horizon is unstable, so the method labels the window 0 (stable). The thresholds $\gamma_{low} = 0.35$ and $\gamma_{high} = 0.65$ are again fixed benchmark-design choices rather than tuned performance parameters. Their role is to exclude mixed target intervals near the transition boundary, so that the supervised task emphasises clearly stable and clearly unstable futures instead of forcing a binary decision on heavily ambiguous cases.

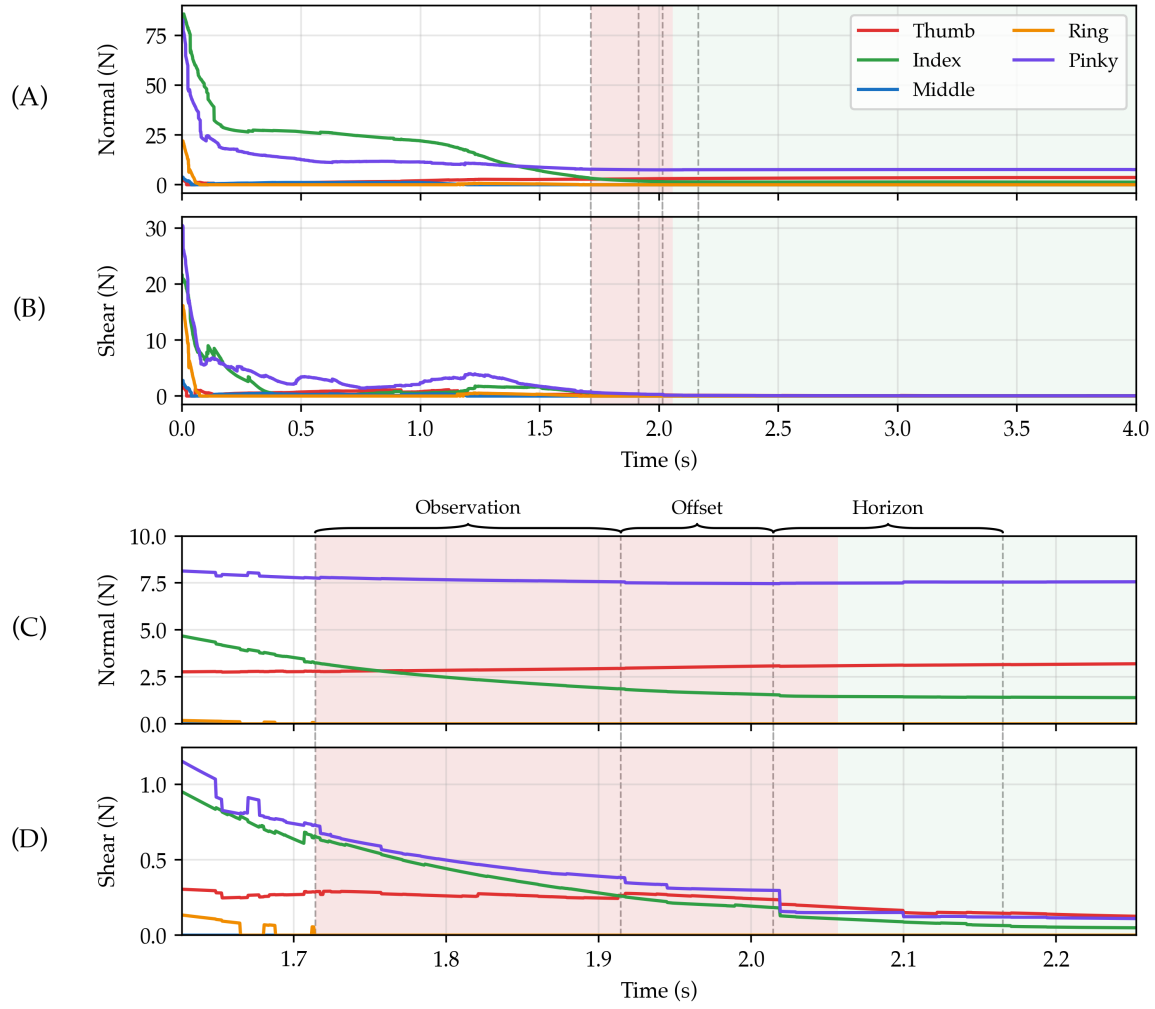


Figure 3.8: Example of one observation window, predictive offset, and future horizon, marked by vertical dashed lines on the replay force data for the apple grasp shown in fig. 3.6a. Red shading marks unstable periods and green shading marks stable periods. In this example, the horizon contains both behaviours, but because only 28.24% of the horizon is unstable, the window receives label 0 (stable). Panels (C–D) zoom in on panels (A–B).

3.4 MODEL ARCHITECTURES

All predictive models discussed in this thesis consume the same normalised input tensor $X_s \in \mathbb{R}^{W\times 5\times C}$ defined in eq. (27). For each window, the model outputs one value z_s describing how strongly that window suggests future instability in the target interval defined in section 3.3.4. The sigmoid function converts this value into a probability,

$$p_s = \sigma(z_s) = \frac{1}{1+e^{-z_s}}, \tag{31}$$

where $p_s \approx 1$ means the model predicts that the grasp will become unstable, and $p_s \approx 0$ means it predicts that the grasp will remain stable. Each sample contains the same observation window regardless of architecture: W timesteps, five fingertips, and $C = 2$ force channels per fingertip in the default setting, corresponding to normal force and shear force. The models do not operate on raw tactile images. They operate on compact force histories derived from the simulation.

Keeping the input representation fixed makes the comparison easier to interpret. The models do not differ in what information they receive. They differ only in how they summarise the force history inside the window. The CNN searches for short local patterns in time, whereas the GRU and LSTM build a running summary of how the grasp evolves. This comparison tests whether short force patterns suffice or whether explicit temporal memory is more useful for this task. Figure 3.9 summarises the comparison before the next subsections describe the architectures.

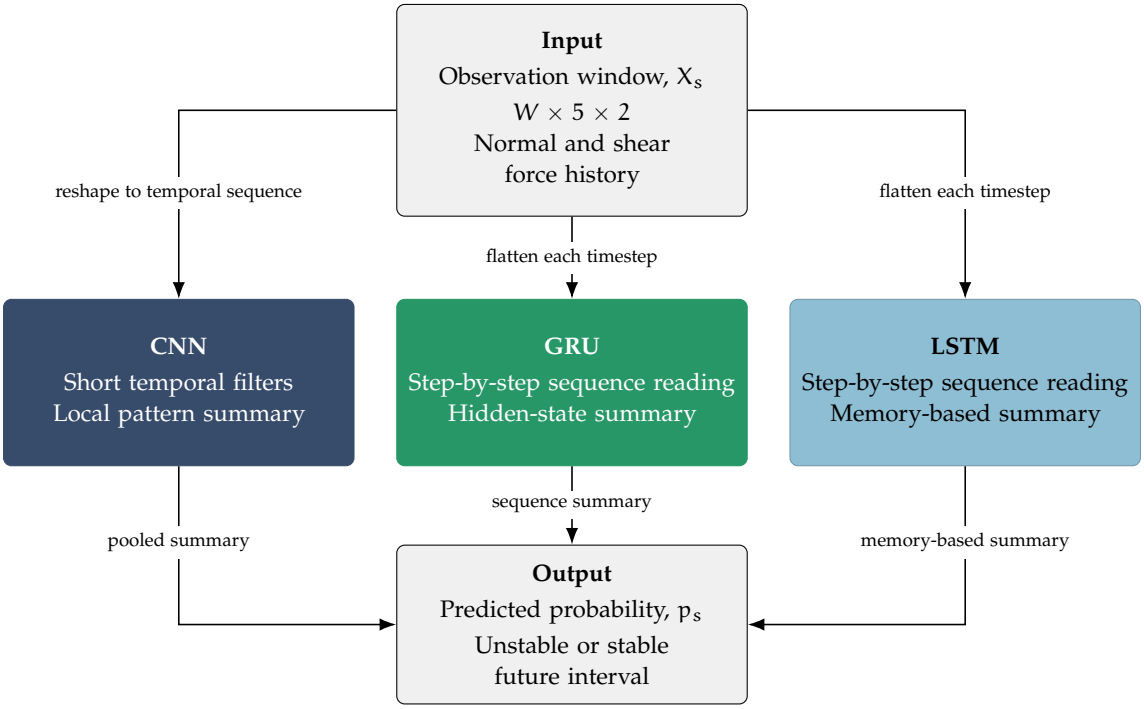


Figure 3.9: High-level comparison of the three predictive model families. Each model receives the same observation window X_s, defined in eq. (27), which contains the recent normal and shear force history. The CNN summarises this window using local temporal filters, whereas the GRU and LSTM read the sequence step by step and build an internal summary over time. All three models then output the probability that the grasp will become unstable in the future target interval defined in section 3.3.4.

3.4.1 Convolutional Neural Network (CNN)

The CNN first rearranges the observation window X_s from eq. (27) into a one-dimensional sequence. It combines the five fingertips and two force channels, so the model receives the full observation window as one sequence with 5C channels at each timestep.

The sequence then passes through three successive one-dimensional convolution layers. These layers use kernel sizes of 5, 5, and 3 respectively. Here, kernel size refers to how many neighbouring

timesteps one convolution examines at once. For example, a kernel size of 5 means the model looks at five consecutive timesteps before moving along the sequence. The full window is still available to the model, but each convolution only looks at one short part at a time. As the filters slide across the whole sequence, the CNN can detect the same kind of pattern wherever it appears in the window. Stacking three convolution layers lets later layers combine and refine patterns detected by earlier ones.

After each convolution, the network applies batch normalisation and GELU activation. The earlier layers also use dropout. During training, dropout randomly ignores some of the values produced within the network, which discourages the model from relying too heavily on one small set of features. Dropout therefore helps the network generalise better to unseen grasps rather than memorising specific details of the training set.

After this, adaptive average pooling compresses the time dimension into one summary vector for the whole window. A small fully connected classifier then converts that summary into the future-instability score z_s.

This architecture matches the central idea introduced earlier in the dissertation: useful warning signs of slip or loss of grasp stability may appear as short local changes in the force history before full instability occurs. In this setting, the CNN looks for these short warning patterns anywhere within the observation window, such as a brief rise in shear force, a short drop in normal force, or a quick shift in how the load shifts across the fingertips.

3.4.2 Gated Recurrent Unit (GRU)

The GRU receives the observation window X_s from eq. (27), which contains W timesteps of normal and shear force history across the five fingertips. Instead of scanning this window for short patterns with convolutions, it reads the sequence one timestep at a time from start to finish. Each timestep becomes a vector in $\mathbb{R}^{5C}$. In the default setting, this means each timestep contains ten values: two force channels for each of the five fingertips.

After reading each timestep, the GRU updates an internal hidden state. This hidden state is the model's running summary of what it has seen so far in the grasp. Because the model carries that summary forward through the sequence, the meaning of later force values can depend on what happened earlier in the window.

After reading the full window, the GRU reduces the sequence to one summary vector and passes it to a small fully connected classifier, which produces the future-instability score z_s.

This architecture matches the case where the order of events matters. A rise in shear force may carry a different meaning if it follows a gradual loss of normal force than if it follows a period of steady loading. In this setting, the GRU tests whether the recent history of the grasp, rather than only short isolated patterns, helps to predict future instability.

3.4.3 Long Short-Term Memory Network (LSTM)

The LSTM also receives the observation window X_s from eq. (27), containing the same W timesteps of normal and shear force history. Each timestep becomes a vector in $\mathbb{R}^{5C}$. In the default setting, this means each timestep contains ten values: two force channels for each of the five fingertips. The LSTM then reads the window one step at a time.

The main difference from the GRU is the way the LSTM stores information. Instead of keeping only one running summary, it keeps both a hidden state and a cell state. The cell state carries information forward over longer parts of the sequence, while the hidden state reflects what is most useful at the current timestep. As the LSTM reads each new timestep, it updates these two states using both the new force values and the information already carried forward from earlier in the window. This gives the model more control over what to keep, what to forget, and what should still influence the prediction later in the sequence.

After processing the full window, the LSTM reduces the sequence to one summary vector and passes it to a small fully connected classifier to produce the future-instability score z_s.

This architecture matches the case where the warning signs of instability build more gradually across the observation window rather than appearing as one short change. In that setting, the LSTM is better placed to hold earlier force trends until later events clarify them.

For the reported window-length experiments, each model family used one tuned configuration across all window sizes. The CNN used 96 base channels. The GRU used two bidirectional recurrent layers with hidden size 64, and the LSTM used one recurrent layer with hidden size 64. In the sweep, both recurrent models used mean pooling across the window before the final classifier. This was a deliberate sweep-level choice rather than an accidental leftover from Optuna: fixing mean pooling for both recurrent models made the comparison more directly about how each model used the full observed history, rather than allowing one recurrent model to rely only on the final hidden state while the other pooled over the whole window. Keeping these settings fixed meant that the window length could vary without changing the underlying model design. All three models share the broader training procedure, which the next section describes.

3.5 TRAINING PROTOCOL AND SPLIT STRATEGY

The CNN, GRU, and LSTM models all use the same training pipeline. The same dataset format, feature normalisation, loss function, optimiser, early-stopping rule, decision-threshold-selection procedure, and evaluation code support all three model families. This keeps the comparison focused on temporal modelling choices rather than differences in training procedure.

3.5.1 Split Strategy

The reported experiments use a multi-object split strategy in which the validation and test sets contain objects unseen during training. For each object, the pipeline generates grasps as described in section 3.1, records replay force data as described in section 3.2, and constructs predictive windows and labels as described in sections 3.3.3 and 3.3.4. During this dataset-construction stage, the pipeline keeps all windows extracted from the same replayed grasp together. This prevents neighbouring windows from the same trajectory leaking across different parts of the experiment.

In total, the reported experiments use data from twelve objects. Eight objects provide training data, two further objects provide validation data, and the remaining two objects serve as the final test set. The models never see the test objects during training. This means the final evaluation measures how well the models generalise to unseen objects, rather than only to unseen grasps of familiar objects.

The final training, validation, and test sets combine all windows from the objects assigned to each of these three groups. The dataset files store the split used for every saved sample, so other researchers can reproduce the experiments.

3.5.2 Normalisation and Training Procedure

Before training, the method normalises the force values so that different fingertips and force channels sit on a comparable numerical scale. Without this step, one feature could dominate optimisation simply because its values take larger numerical values, rather than because it carries more useful information. In this work, the method computes the normalisation statistics from the training split only. For each fingertip and each force channel, it estimates one mean and one standard deviation using all training windows and all timesteps in the training set. It then applies these same training-set statistics unchanged to the validation and test sets. If x denotes a raw feature tensor, the normalised tensor becomes

$$\tilde{x} = \frac{x - \mu_{\text{train}}}{\sigma_{\text{train}}}, \tag{32}$$

where μ_{train} and σ_{train} denote the training-set mean and standard deviation. Both come from the training split only. In practice, the implementation applies a small floor ε to the denominator so that division by zero cannot occur. This prevents information from validation or test objects leaking into the preprocessing stage.

The pipeline then trains all three models as binary classifiers. For each observation window, the model outputs the raw score z_s introduced above, and the training loop compares this score with the true label using binary cross-entropy with logits,

$$\mathcal{L}(z_s, y_s) = \text{BCEWithLogits}(z_s, y_s), \tag{33}$$

where z_s is the raw score before the sigmoid transformation and $y_s \in \{0, 1\}$ is the true label. This loss combines two steps into one stable calculation: it first converts the raw score into a probability, and then measures how well that probability matches the true stable or unstable label. The loss is small when the model gives unstable windows high probabilities and stable windows low probabilities, and it becomes larger when the model is confidently wrong. Using the raw score directly in this combined form is more numerically stable during training than applying the sigmoid separately. Optimisation uses the AdamW optimiser. For the reported window-length sweep, each model family used one tuned configuration across all window sizes. This means that settings such as batch size, learning rate, weight decay, dropout, and the relative penalty assigned to unstable examples can differ slightly between the CNN, GRU, and LSTM, but the training loop itself remains the same.

Training runs for at most 40 epochs. After each epoch, the method evaluates the current model on the validation set. If its validation performance is the best seen so far, the method saves that version of the model. Training can then continue, but if no later epoch performs better on the validation set, the method uses the saved version for the final evaluation. Early stopping ends training after the validation metric has failed to improve for the configured number of consecutive epochs, referred to as the patience. For the reported sweep, the method keeps the model with the highest AUPRC on the validation set for that seed.

AUPRC stands for area under the precision–recall curve. The metric uses the full set of validation windows, not a single window. After an epoch, the same trained model scores every window in the validation set. For each window, it outputs the predicted probability of future instability in the target interval. The method then orders these windows from highest predicted instability probability to lowest. If windows that truly lead to future instability tend to appear near the top of this ordering, the AUPRC increases.

The two terms in the name describe the measurement. Precision asks: among the windows given high instability probabilities, how many really do lead to future instability? Recall asks: among all windows that really do lead to future instability, how many receive high instability probabilities from the model? AUPRC combines these two ideas into one number.

The AUPRC metric fits the practical aim of the project. The model should assign high probabilities to windows that really do lead to future instability, without assigning similarly high probabilities to too many windows that remain stable. That makes AUPRC more informative for model selection than accuracy, which can still look strong even when the model does not separate unstable and stable windows well.

3.5.3 Decision-threshold Selection

The models output probabilities, so the method must choose a decision threshold before it can label each window as stable or unstable. This decision threshold is the probability boundary that converts the predicted instability probability into a binary decision. A fixed decision threshold of 0.5 is the simplest option, but this work does not assume in advance that 0.5 is the best choice. Instead, after the method selects the best model for a given seed, it uses the validation set to choose the final decision threshold. In practice, this means trying many candidate decision thresholds on the validation predictions and keeping the one that best matches the aim of the task. This matters because the task is not symmetric: missing an unstable grasp is generally more costly than issuing a conservative warning for a grasp that would have remained stable.

For each seed, the method tests 181 candidate decision thresholds between 0.05 and 0.95 on the validation set. The chosen decision threshold is the one that gives the highest validation precision while still keeping validation recall at or above 0.9. If none of the candidate decision thresholds meets this recall target, the method uses the decision threshold with the best validation F_1-score instead. The test split is not used during this process. The method evaluates it only after the validation predictions have already fixed the decision threshold. This search uses a dense validation-only

grid, with values spaced every 0.005 across the interval from 0.05 to 0.95. In practice, this is fine-grained enough to approximate continuous decision-threshold tuning while avoiding degenerate edge decision thresholds near 0 and 1. The 0.9 minimum-recall target reflects the asymmetry of the task: missing a future unstable interval is treated as more costly than issuing a conservative warning for one that would have remained stable.

This protocol reflects the practical aim of the project: raising some conservative warnings is better than missing a grasp that is about to become unstable.

3.5.4 Multi-seed Evaluation and Controlled Sweeps

The study repeats each training setting over five random seeds, namely 42 to 46. For every seed, it repeats the full procedure: model initialisation, optimisation, early stopping, selection of the saved model from the best validation epoch, selection of the final decision threshold, and final validation and test evaluation. The reported results use the mean and standard deviation across seeds rather than one favourable run. This gives a more reliable comparison by showing how sensitive each setting is to random initialisation and data ordering.

The evaluation records loss, accuracy, precision, recall, F_1-score, AUROC, AUPRC, and the entries of the confusion matrix. The checkpoint from the best-performing seed is also saved, but the comparative analysis uses the multi-seed summaries.

To compare the effect of window length, the window-length sweep keeps the end time of each sample fixed across all window sizes. The method first generates a reference dataset at the longest window length in the sweep. It then creates shorter windows by trimming frames from the start of each reference window. As a result, each sample keeps the same end time, the same predictive offset, the same future target interval, the same label, and the same split membership. The number of samples in each split also remains fixed across window lengths. The only change is the amount of past force history the model receives.

This makes the window-length sweep a controlled comparison of how much recent force history is useful for prediction. Differences in performance then reflect the amount of recent force history available, rather than changes in sample identity, label definition, or split composition.

4 EVALUATION AND DISCUSSION

Chapter 3 described a full pipeline for predictive grasp-stability learning from replayed fingertip force histories. This chapter asks three related questions: whether that formulation produces useful predictive performance on objects not seen during training, what it implies for model choice in practice, and how the resulting signal can be interpreted physically. It evaluates the proposed models on a shared unseen-object benchmark, then examines how robust the conclusions remain across split choice, observation-window length, and latency. It finally returns to the friction-cone discussion from section 2.2 by comparing the learned models with a ratio-based proxy based on that idea.

4.1 EXPERIMENTAL SETUP

4.1.1 Split Composition

The evaluation uses the multi-object split described in sections 3.2 and 3.3.3. Eight YCB objects provide the training data: foam brick, gelatin box, lemon, mug, mustard bottle, master chef can, pudding box, and sponge. Two further objects, apple and potted meat can, provide the validation set used for early stopping and decision-threshold selection. The final test set contains the remaining two objects, pear and Rubik's cube. After window construction, the main comparison uses 43,978 training windows, 12,052 validation windows, and 14,587 test windows. The reported test results therefore measure transfer to unseen objects, not only to unseen replays of familiar objects.

Each split contains a substantial number of both label classes. The training set contains 11,538 stable and 32,440 unstable windows, the validation set 4,152 stable and 7,900 unstable windows, and the test set 5,955 stable and 8,632 unstable windows. The unstable class is more common throughout, but the stable class is still large enough for a useful comparison between the two classes.

In this chapter, each sampled observation window is treated as one evaluation example, rather than treating the whole grasp as one example. A single grasp can therefore contribute several overlapping windows, but all windows from that grasp stay in the same split. The reported metrics are therefore window-level rather than grasp-level.

4.1.2 Tuning and Evaluation Protocol

To compare the prediction-model architectures fairly, one fixed prediction task is used. Each model observes a 0.20 s window and predicts behaviour over the future interval defined in section 3.3.3: a 0.10 s offset followed by a 0.15 s horizon. The comparison then uses one saved baseline configuration for each model family. These are the same multi-object configurations used in the CNN, GRU, and LSTM notebooks. Table 4.1 summarises the broader Optuna search space used later in the chapter.

Each baseline configuration is trained over five random seeds (42–46). For each seed, the best checkpoint is chosen by validation AUPRC. The final decision threshold is then chosen on the validation set using the precision-at-minimum-recall rule described in sections 3.5.3 and 4.2. The test set is used only after these choices are fixed.

Setting group	Settings varied during Optuna search
Optimisation	Learning rate, weight decay, batch size, and the gradient-clipping limit used to keep training stable.
Regularisation and stopping	Dropout, patience, and the minimum improvement required for early stopping.
Threshold and class balance	Validation selection metric, decision-threshold rule, minimum recall target, and whether the loss reweights unstable examples according to the class frequencies in the training set.
Architecture-specific settings	CNN: base channels. GRU & LSTM: hidden size, number of recurrent layers, whether the sequence is processed in one or both time directions, the way the sequence is summarised before classification, and recurrent dropout.

Table 4.1: Optuna search space used for the later tuned short-window and controlled-sweep studies.

The three baseline configurations use similar training settings and mainly differ in model size. Table 4.2 summarises the settings and trainable parameter counts. These counts matter because the model families are not capacity-matched. In the rest of the chapter, validation results are used for model and window selection. The test set is reserved for the final evaluation on unseen objects, after the model and decision-threshold choices have already been fixed on the validation set.

Model	Parameters	Architecture settings	Training settings
CNN	178,753	Base channels: 64	Dropout: 0.2, learning rate: 10^{-3}, batch size: 256, weight decay: 10^{-4}, patience: 10
GRU	171,777	Hidden size: 128, recurrent layers: 2, direction: unidirectional, temporal pooling: mean	Dropout: 0.2, learning rate: 10^{-3}, batch size: 256, weight decay: 10^{-4}, patience: 10
LSTM	25,217	Hidden size: 64, recurrent layers: 1, direction: unidirectional, temporal pooling: mean	Dropout: 0.2, learning rate: 5×10^{-4}, batch size: 256, weight decay: 10^{-4}, patience: 10

Table 4.2: Baseline configurations for the fixed-task architecture comparison at the baseline 0.20 s observation window.

4.2 PERFORMANCE METRICS

Because the classes are uneven, the evaluation reports more than accuracy. Two groups of metrics are used. The ranking metrics are AUPRC and AUROC. These test how well a model gives higher scores to risky windows than to safe ones across all possible decision thresholds. In this dissertation, AUPRC is the main ranking metric because it focuses on the unstable class, which is the class of practical interest.

The decision-threshold metrics are precision, recall, F_1-score, and accuracy. These are reported after choosing one validation-selected decision threshold, as described in section 3.5.3. Precision measures how often a warning is correct. Recall measures how many truly unstable future intervals are flagged. The F_1-score combines precision and recall into a single value, so it summarises the balance between them. Accuracy is also reported, but it is secondary here.

The decision-threshold rule reflects the asymmetry of the task. Missing an unstable future interval is usually worse than issuing a conservative warning. For that reason, the method chooses the most precise decision threshold that still reaches at least 90% recall on the validation set. If no decision threshold reaches that level, it falls back to the validation decision threshold with the best F_1-score.

In short, the ranking metrics judge how well the score ordering works, whereas the decision-threshold metrics judge the quality of one chosen operating point. The fixed-window comparison

asks which model is strongest on one shared benchmark. The later window-length and latency analysis asks which architecture–window pair gives the most practical operating point.

4.3 FIXED-TASK BENCHMARK AT THE BASELINE OBSERVATION WINDOW

At the baseline 0.20 s observation window, all three architectures perform well on the shared benchmark task. The GRU gives the strongest overall benchmark result.

Figure 4.1 shows that all three models separate stable and unstable futures well on the validation objects. The ranking results in fig. 4.1C are very close. The GRU reaches the highest mean validation AUPRC at $99.25 \pm 0.01\%$, with the LSTM and CNN close behind at $99.24 \pm 0.01\%$ and $99.06 \pm 0.06\%$. The decision-threshold metrics in fig. 4.1A are also close: the GRU reaches $94.49 \pm 0.45\%$ validation F_1, the CNN $94.16 \pm 0.55\%$, and the LSTM $93.80 \pm 0.03\%$.

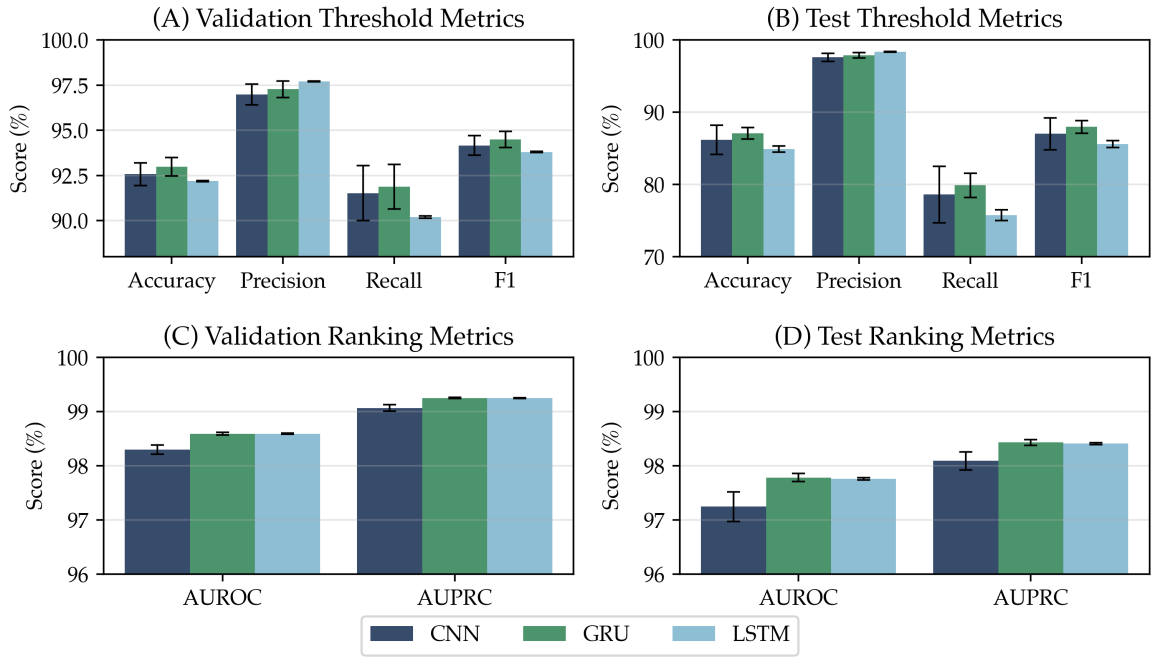


Figure 4.1: Validation and test performance of CNN, GRU, and LSTM at the baseline 0.20 s observation window. Figure 4.1A shows validation accuracy, precision, recall, and F_1-score at the validation-selected decision threshold. Figure 4.1B shows the corresponding test metrics. Figure 4.1C shows validation AUROC and AUPRC. Figure 4.1D shows test AUROC and AUPRC. Bars show the mean across five seeds, with error bars denoting ± 1 standard deviation.

The unseen-object test split gives the clearer answer. The ranking metrics in fig. 4.1D again place the GRU slightly ahead, with $98.43 \pm 0.05\%$ AUPRC and $97.78 \pm 0.07\%$ AUROC. The decision-threshold metrics in fig. 4.1B are closer together. All three models still produce a strong operating point once the validation-selected decision threshold has been fixed. The GRU achieves the strongest mean F_1-score through slightly higher recall, while the LSTM reaches the highest precision. The CNN remains competitive, but with larger variation across seeds.

The gap between the models is modest, not large. On the fixed 0.20 s task, the GRU is the strongest benchmark model. The gap is small. The comparison is also not capacity-matched: the LSTM is much smaller than the other two models and still stays close in performance. The evidence therefore suggests only a modest advantage for explicit recurrent memory.

The seed-to-seed variation supports that reading. The CNN remains close in mean performance, but its test-set standard deviation is larger than that of the recurrent models, especially on F_1-score. The GRU and LSTM therefore seem more stable across random initialisation. Much of the useful

signal appears to lie in short recent temporal patterns that every model can use. Sequence-aware modelling then helps the GRU rank future instability slightly more reliably.

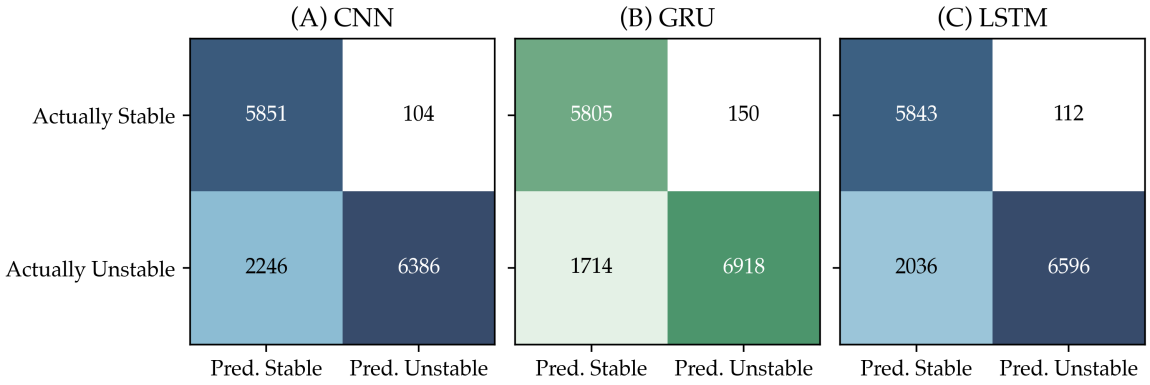


Figure 4.2: Illustrative test-set confusion matrices for seed 42 at the baseline 0.20 s observation window. Figure 4.2A shows the CNN, fig. 4.2B the GRU, and fig. 4.2C the LSTM. Rows indicate the true class and columns the predicted class. Cell values are normalised within each row.

Figure 4.2 shows the same pattern more concretely. These single-seed matrices do not replace the five-seed summaries, but they make the errors easier to see. All three models produce far fewer false positives than false negatives on the unseen test objects. This matches the validation decision-threshold rule. The main remaining error after transfer is therefore missed unstable windows rather than false alarms.

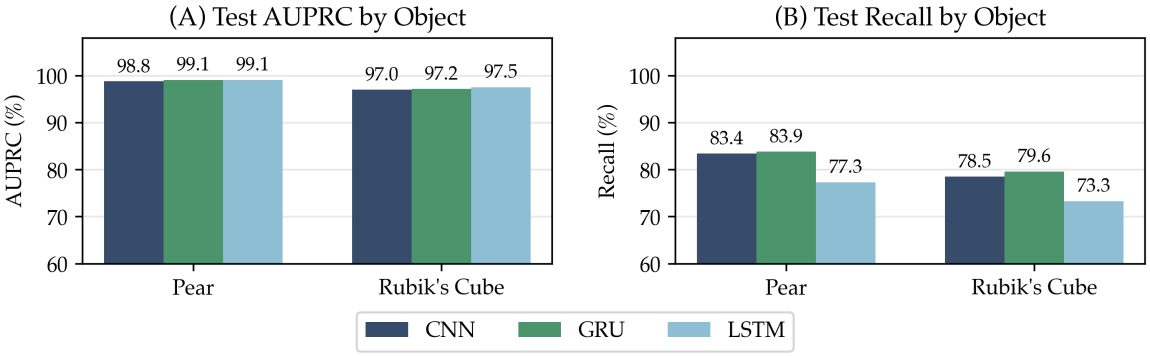


Figure 4.3: Per-object diagnostic test metrics for the saved CNN, GRU, and LSTM checkpoints at the baseline 0.20 s observation window. Figure 4.3A shows test AUPRC for pear and Rubik's cube. Figure 4.3B shows test recall at the validation-selected decision threshold for the same two unseen test objects.

The pooled test metrics also hide an asymmetry across the two unseen test objects. Figure 4.3 shows that pear is the easier unseen test object for all three model families, whereas Rubik's cube is consistently harder. The AUPRC gap is modest, so all three models still rank risky windows above safe ones well on both objects. The difference is clearer in recall, where every model recovers a smaller fraction of unstable windows on Rubik's cube. The pear–Rubik's cube difference does not alter the benchmark verdict, but it does show that transfer is not equally easy on every unseen object.

Because the repository currently stores one selected checkpoint per model family rather than one checkpoint per seed, fig. 4.3 should be read as a diagnostic view only. It does not replace the five-seed benchmark. The diagnostic is useful because it shows object-level structure hidden by the pooled test averages. Overall, the benchmark shows strong transfer from force histories, with the GRU narrowly ahead at this setting.

4.4 SPLIT ROBUSTNESS ACROSS UNSEEN–OBJECT PAIRINGS

The fixed split used throughout the chapter is only useful if it is reasonably typical. This section therefore asks whether that split is unusually easy or unusually favourable.

With twelve objects, there are 2970 ways to choose the validation and test object pairs when the two roles are kept distinct. Swapping one pair from validation to test therefore counts as a different split. If the validation and test roles are ignored, this becomes 1485 distinct unseen-object pair combinations.

Running the full training procedure on all of these assignments would be expensive and unnecessary here. The study therefore samples 30 distinct validation-test assignments, including the fixed split used throughout the chapter.

The split-robustness analysis uses the Optuna-optimised LSTM selected for the short-window setting (0.01 s) and repeats each sampled assignment over three seeds. The resulting experiment is a lighter robustness check, not a second full benchmark. For simplicity, fig. 4.4 shows the LSTM only. The other model families follow the same broad pattern.

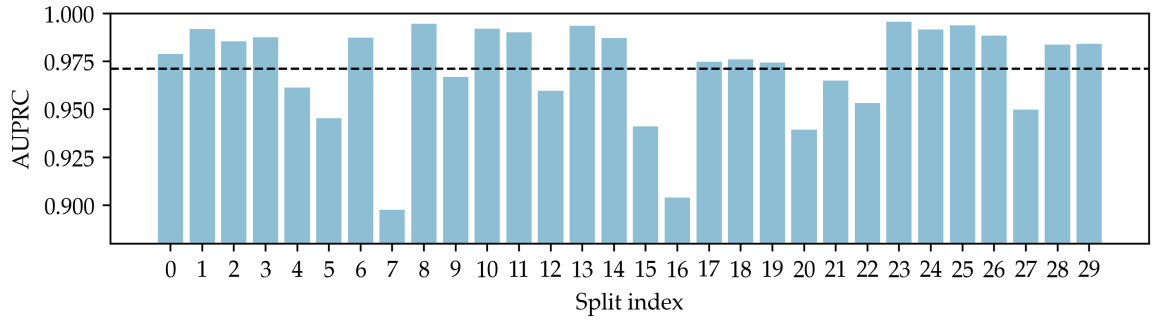


Figure 4.4: Split-robustness diagnostic for the 0.01 s LSTM configuration. Each bar shows test AUPRC for one of the 30 sampled unseen-object split assignments, and the dashed line marks the mean across all 30 assignments.

Figure 4.4 suggests that the fixed split is not unusually favourable. Across the 30 sampled assignments, the LSTM reaches a mean test AUPRC of $97.14 \pm 2.49\%$, with a median of 98.13%. Every sampled split stays above 90% AUPRC, and many lie close to 99%. On the fixed split, the same 0.01 s LSTM reaches 97.90% test AUPRC. That places it near the centre of the sampled distribution.

The lower-scoring assignments are informative as well. Among the 30 sampled assignments, the hardest unseen-object pairings repeatedly involve `foam_brick`, especially when combined with `lemon` or `sponge`. Transfer is therefore not equally easy for every unseen object combination. Even so, the main conclusion is reassuring: the broader predictive signal does not depend on one especially convenient partition.

The split-robustness result matters for the rest of the chapter because section 4.5 keeps using the same fixed split. The present check suggests that this split is fairly typical rather than unusually easy, so the later comparisons are less likely to benefit from one favourable partition.

4.5 WINDOW LENGTH AND PRACTICAL OPERATING CHOICE

The fixed benchmark above uses a baseline observation window of 0.20 s. This section asks whether that much recent history is actually needed. The sweep suggests that similar predictive performance can be achieved with much shorter windows.

The study repeats the experiment across window lengths from 0.01 s to 0.50 s while keeping the predictive offset, future horizon, split membership, and sample end time fixed, as described in section 3.3.3.

Unlike the fixed-task comparison, this sweep does not re-tune every architecture at every window length. Instead, one tuned configuration per model family is kept fixed across the whole sweep. Changes can therefore be attributed to window length rather than to changes in model size or

optimiser settings. For the recurrent models, the sweep also fixes mean pooling for both the GRU and LSTM.

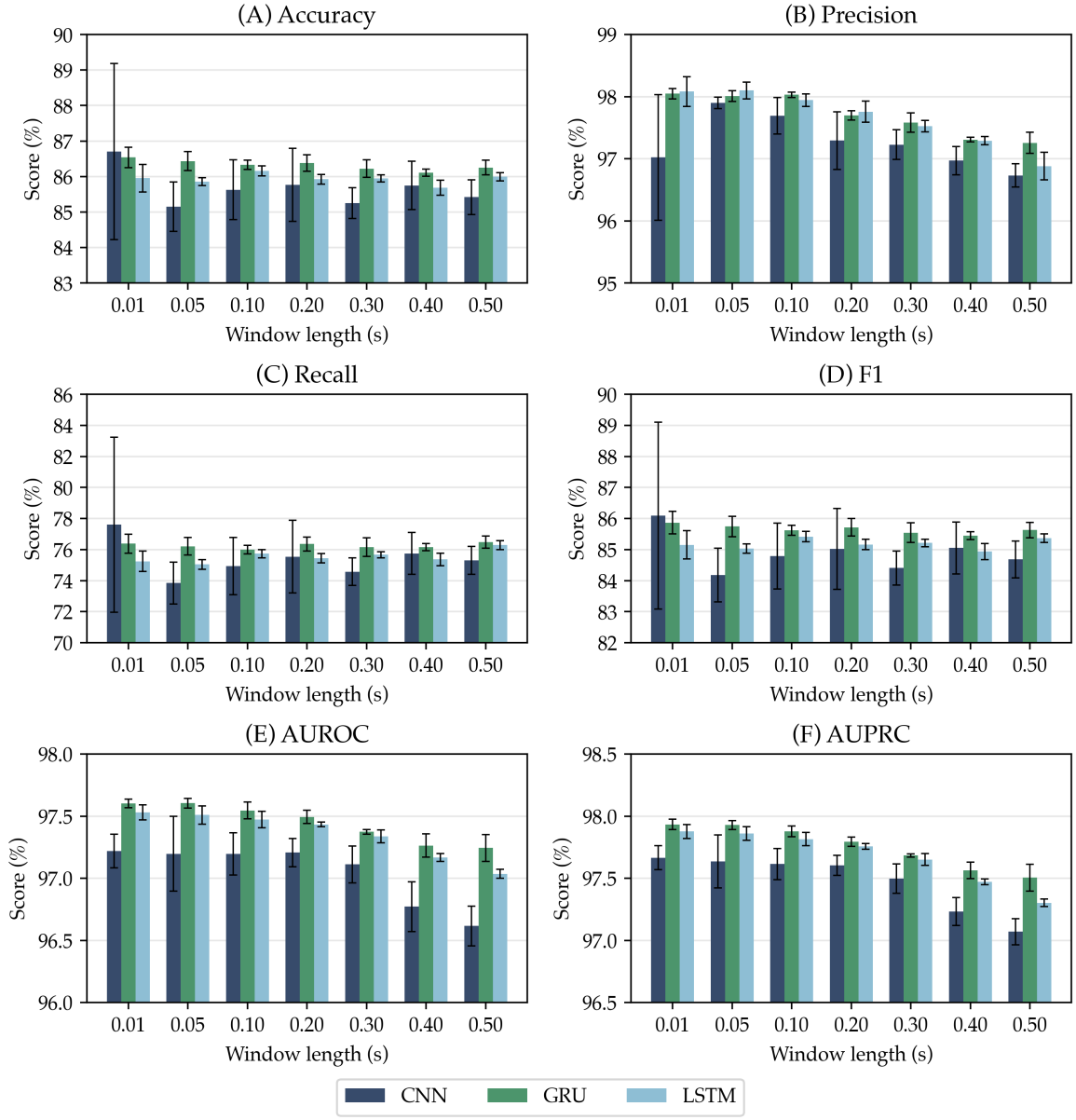


Figure 4.5: Test metrics across observation-window length for CNN, GRU, and LSTM. Figure 4.5A–F show accuracy, precision, recall, F_1-score, AUROC, and AUPRC respectively. Bars show the mean across five seeds, with error bars denoting ± 1 standard deviation. The predictive offset and future horizon remain fixed across the sweep.

Figure 4.5 shows only modest performance changes across this range. In fig. 4.5D, the CNN stays between 84.2% and 86.1% mean test F_1, the GRU between 85.4% and 85.9%, and the LSTM between 85.0% and 85.4%. The ranking metrics in fig. 4.5E and fig. 4.5F show the same broad pattern. AUPRC and AUROC stay high throughout the sweep and decline only gradually as the window becomes longer. There is no evidence here that half a second of force history improves prediction much over much shorter windows.

The strongest mean F_1 occurs at 0.01 s for both the CNN and GRU, and at 0.10 s for the LSTM. These differences are small, but the pattern is consistent. The most informative cues appear to occur close to the prediction point. The models seem to benefit most from the most recent 10–100 ms of force history. Longer windows provide older context, but that older context does not produce a stronger predictive signal than the short history immediately preceding the target interval.

This also matches the labelling design in sections 3.3.3 and 3.3.4. Each sample is labelled from behaviour that begins only 0.10 s after the observation window and continues for a short 0.15 s horizon. Under that design, the most useful cues would be expected to appear close to the prediction point rather than much earlier in the grasp. The sweep therefore supports a simple conclusion: short recent histories contain most of the useful signal for this task.

For practical use, predictive performance is only part of the picture. A warning model must also run quickly enough for a real-time control loop. Two latency measures matter here. The first is forward-pass latency: how long the model takes once the observation window is already available. The second is total decision latency, which also includes the time needed to observe the full window.

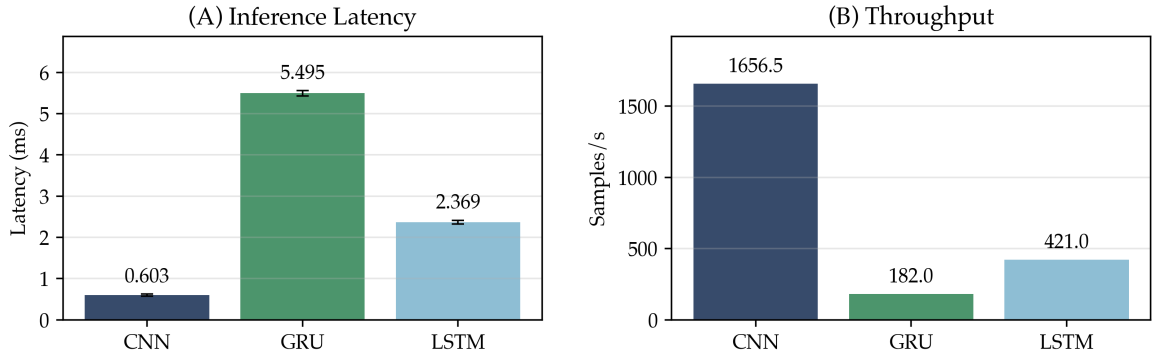


Figure 4.6: CPU forward-pass latency of the original multi-object comparison checkpoints at batch size 1 and the baseline 0.20 s observation window. Figure 4.6A shows median inference latency per window. Figure 4.6B shows throughput in samples per second. Error bars denote standard deviation across repeated forward passes.

Measured on the same device, the three checkpoints show a consistent runtime ordering in fig. 4.6. The CNN is the fastest model at 0.625 ± 0.032 ms per window. The LSTM follows at 2.407 ± 0.048 ms, and the GRU is slowest at 5.634 ± 0.118 ms. fig. 4.6B shows the same ordering in throughput, at roughly 1600, 415, and 178 predictions per second. Measured only as forward-pass computation, all three models remain comfortably below a 10 ms control-cycle budget on CPU.

This runtime gap should not be read as a general statement about GRUs and LSTMs. Here it reflects the specific checkpoints used in the benchmark comparison. The GRU uses hidden size 128 and two recurrent layers, whereas the LSTM uses hidden size 64 and one recurrent layer. The GRU therefore performs more sequential computation per window. The CNN remains faster still because its temporal convolutions are more efficient once the input window is available.

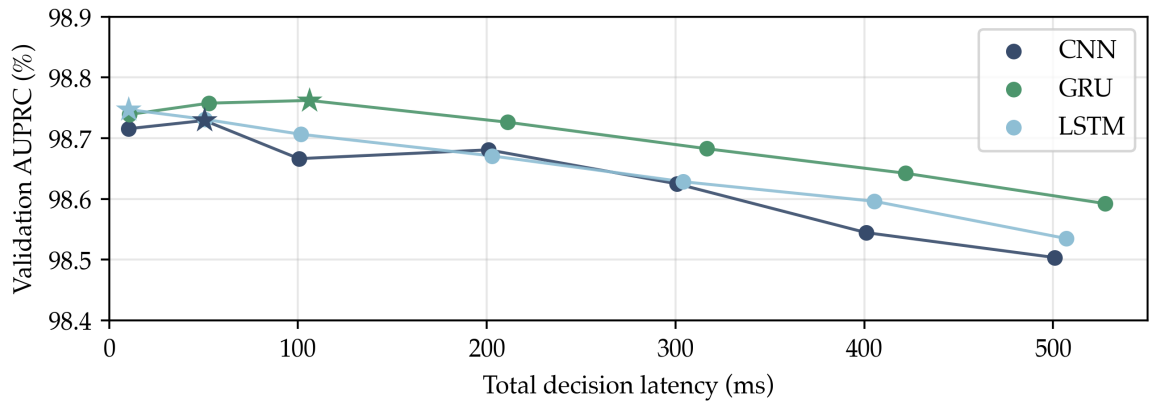


Figure 4.7: Validation AUPRC against total decision latency across the window-length sweep. Total decision latency equals observation-window duration plus same-device CPU forward-pass inference latency at batch size 1. Within each model family, the star marks the window length that gives that family's highest mean validation AUPRC.

Forward-pass latency is only part of the deployment trade-off, because the system must first observe the whole window. Figure 4.7 therefore compares validation AUPRC with total decision latency across all architecture–window pairs. In each model family, the star marks the window length with the highest mean validation AUPRC. The pattern is simple: longer windows add delay almost linearly, but the gains in validation AUPRC are very small.

The highest mean validation AUPRC occurs at the GRU with a 0.10 s window, reaching $98.762 \pm 0.034\%$. That setting, however, has a total decision latency of 106.10 ms.

Using the latency-aware heuristic described above, the preferred operating point becomes the LSTM with a 0.01 s window. That configuration reaches $98.747 \pm 0.022\%$ validation AUPRC with a total decision latency of 10.17 ms from the start of the observation window to a model output. The GRU at 0.01 s stays very close in ranking performance at $98.739 \pm 0.022\%$, but is slightly slower at 10.64 ms.

The latency-aware heuristic is pragmatic rather than statistically optimal. It is only meant to pick a low-latency setting whose validation AUPRC is effectively the same as the best result in the sweep. Under that rule, the short-window LSTM becomes the practical recommendation. Under the fixed benchmark, the GRU remains the strongest model. The two conclusions answer different questions.

For context, reactive human grip-force responses to unexpected distal loads in a standard precision-grip posture have been reported at about 62 ± 9 ms for distal loads and 74 ± 9 ms for proximal loads [46]. The comparison is not like-for-like, because the present latency excludes sensing, communication, and actuation delays. Even so, the comparison places the selected 0.01 s operating point on a useful timescale. More importantly, it reinforces the main point: longer windows add delay, but give little extra predictive value.

4.6 FAILURE CASES, DECISION-THRESHOLD TRANSFER, AND CALIBRATION

Table 4.3 shows the main error pattern for the baseline 0.20 s comparison. Across all three architectures, missed unstable futures are much more common than false alarms. The table uses the same-end reference set employed in the window-length sweep, so its class counts differ slightly from the full test-set counts reported earlier.

Model	False positives	False negatives	False-positive rate	Miss rate
CNN	149 ± 31	1723 ± 165	2.44%	24.46%
GRU	127 ± 4	1666 ± 32	2.07%	23.64%
LSTM	122 ± 10	1730 ± 21	2.00%	24.55%

Table 4.3: Aggregate test errors for the baseline 0.20 s observation window. Values show the mean $\pm$ standard deviation across five seeds. The false-positive rate is normalised by the 6113 stable windows and the miss rate by the 7045 unstable windows in the same-end reference set.

Across all three architectures, false negatives outnumber false positives by roughly an order of magnitude. The models therefore remain conservative on unseen test objects: when they issue a warning it is usually correct, but some unstable futures are still not flagged early enough. These are still window-level errors rather than grasp-level warnings, but they already show clearly where the present formulation remains difficult.

The table also shows the small differences between architectures. The LSTM produces the fewest false alarms, which matches its slightly higher operating precision. The GRU misses the fewest unstable futures, which matches its slightly higher recall and F_1-score on the same split. The CNN sits between these two behaviours. The differences are small, but they support the earlier interpretation that the recurrent models produce slightly cleaner and more stable operating points while the CNN remains competitive.

A plausible explanation is that some replay windows still look like stable loading until very near the transition. That leaves only weak predictive cues inside the observation window. The horizon-based labelling scheme also makes boundary cases difficult. Windows near the transition can differ

only subtly in how much of their future interval is unstable, especially when instability develops gradually or contact ends abruptly. The smaller number of false positives likely reflects the opposite case. A brief rise in shear force or a short redistribution of load can look risky, yet the grasp still re-stabilises within the labelled future horizon.

The main difficulty is therefore not score ordering, but transfer of the validation-selected decision threshold to unseen objects. Test AUPRC remains high across all three models, so they still order risky windows above safe ones well. The problem is that the validation-selected decision threshold, defined in section 3.5.3, becomes slightly too conservative after transfer. That mainly reduces recall while leaving precision high. In practical terms, the models often identify which windows are relatively more dangerous, but still fail to issue a warning early enough in some unseen-object cases.

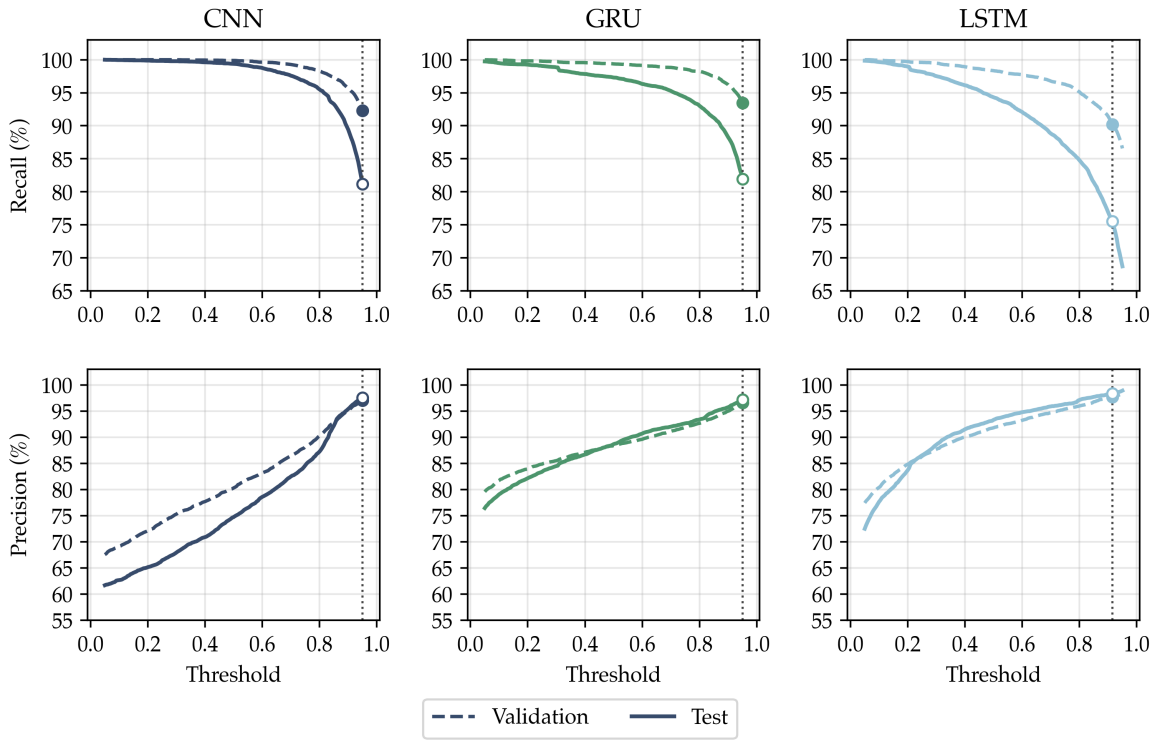


Figure 4.8: Diagnostic for transfer of the validation-selected decision threshold for the saved CNN, GRU, and LSTM checkpoints at the baseline 0.20 s observation window. The top row shows recall against threshold for validation and test, and the bottom row shows precision. The dotted vertical line marks the decision threshold chosen on the validation set and then applied to the test set.

The curves in fig. 4.8 make the same effect easier to see. The dotted vertical line marks the decision threshold chosen on the validation set. Moving the line to the left lowers the decision threshold, which usually increases recall but also creates more false alarms. Moving it to the right raises the decision threshold, which usually improves precision but misses more unstable windows. At the transferred validation-selected line, all three models keep very high test precision, but test recall is lower than on validation. That threshold-transfer pattern explains why high AUPRC can coexist with weaker recall at one chosen operating point: the model can still order windows well even when the transferred decision threshold is slightly too strict for the unseen test objects. The effect is strongest for the LSTM, though it appears for the CNN and GRU as well. That conservative threshold transfer is one reason the practical recommendation in section 4.5 remains cautious.

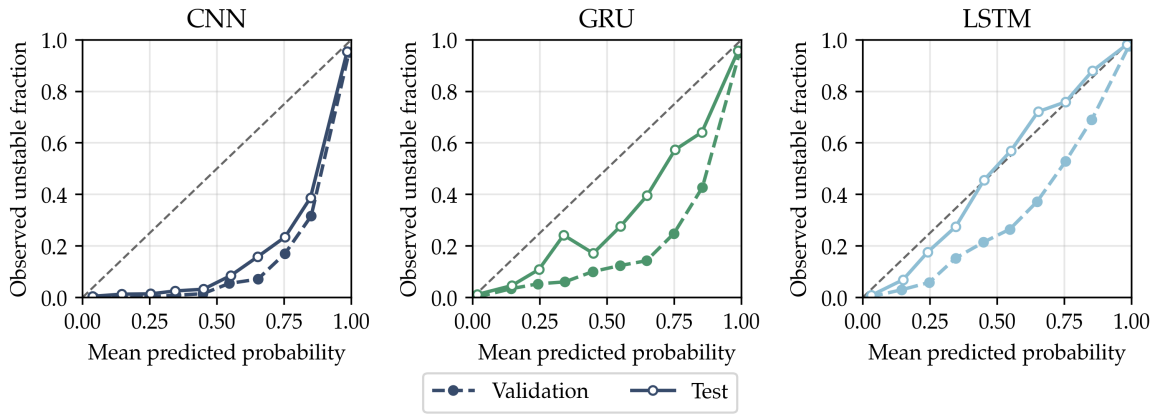


Figure 4.9: Reliability plots for the saved CNN, GRU, and LSTM checkpoints at the baseline 0.20 s observation window. Each panel compares mean predicted instability probability with the observed unstable fraction on validation and test. The diagonal indicates perfect calibration.

Figure 4.9 addresses a different question from ranking performance: whether the predicted probabilities themselves can be read as risk levels. If a curve lies close to the diagonal, a window given a predicted instability probability of, for example, 0.8 is unstable about 80% of the time, so the probabilities are well calibrated. A curve below the diagonal means the model is overconfident, and a curve above the diagonal means it is underconfident. Calibration matters separately from ranking, because a model can still sort risky windows well even when its probability scale is misleading. In this single-checkpoint view, the LSTM follows the diagonal more closely than the other two models. The GRU remains reasonably well calibrated while still giving the strongest ranking performance, whereas the CNN shows the largest mismatch between predicted and observed instability. Calibration therefore adds information beyond AUPRC and F_1: it shows whether the scores themselves can be interpreted directly as risk levels.

The threshold-transfer and calibration diagnostics suggest that fingertip normal and shear forces capture much of the predictive signal, but not all of the object-specific detail that determines whether a local disturbance becomes full grasp failure. That limitation points to a weakness in the current representation. Because the model sees aggregated fingertip forces rather than spatial tactile structure, subtle redistributions of contact may be compressed away before prediction begins.

4.7 FRICTION-CONE INTERPRETATION

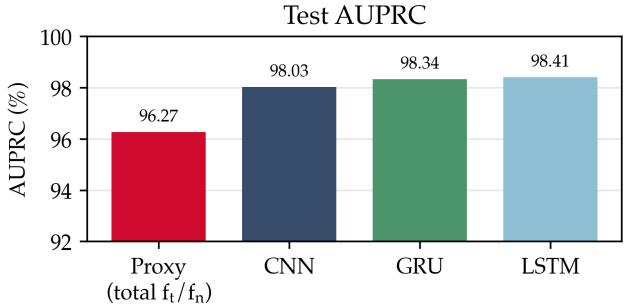


Figure 4.10: Test AUPRC on the unseen-object split for the friction-ratio proxy and the saved CNN, GRU, and LSTM checkpoints at the baseline 0.20 s observation window.

The contact-mechanics discussion in section 2.2 introduced the friction-cone condition: slip becomes more likely when shear force grows too large relative to the available normal force. Figure 4.10 makes that link more concrete by comparing the learned models with a friction-ratio proxy. For

each observation window, the proxy assigns one score: the ratio of total tangential to total normal fingertip load at the final observed timestep. Higher values mean that the grasp is supporting more sideways load for the same amount of squeezing load, so the contact is closer to sliding.

This score is not itself a probability. To evaluate it, a decision threshold is swept across the score: windows above the threshold are predicted unstable and windows below it are predicted stable. The precision and recall produced by all such thresholds define the precision-recall curve, and the area under that curve gives the AUPRC. It should therefore be read as a proxy rather than as an exact friction-cone test. A true friction-cone check would ask, at each fingertip contact, whether the tangential load exceeds the limit set by the local normal load and the friction coefficient μ. The proxy used here instead pools load across fingertips into one ratio and omits μ.

On the unseen-object test split, the proxy reaches 96.27% AUPRC. This makes the previous point about the force representation more concrete: aggregated normal and shear forces already capture a large part of the predictive signal. The learned models still perform better, with test AUPRC rising to 98.03% for the CNN, 98.34% for the GRU, and 98.41% for the LSTM. That remaining gap is also informative. It suggests that the networks are using short temporal patterns in the force history that a single final-timestep ratio cannot capture.

4.8 CHAPTER SUMMARY

The chapter supports three main conclusions. First, predictive grasp-stability modelling from sensor-invariant fingertip force histories generalises well to unseen objects in simulation. On the baseline benchmark, the GRU reaches $98.43 \pm 0.05\%$ test AUPRC and $97.78 \pm 0.07\%$ test AUROC, while the CNN and LSTM remain close behind. A friction-ratio proxy also reaches 96.27% test AUPRC on the same split, suggesting that much of the predictive signal is already grounded in the relation between tangential and normal load, although the learned models still perform better. That conclusion is strongest at the level actually tested here: sampled windows from unseen-object replays.

Second, on the baseline 0.20 s benchmark, the GRU is the strongest model. Its mean validation AUPRC reaches $99.25 \pm 0.01\%$, but the margin over the CNN and LSTM is small and the comparison is not capacity-matched. The most cautious reading is that sequence-aware modelling helps, but only modestly.

Third, once observation time is counted as part of total decision latency, very short windows become more attractive. Under the chosen validation-based heuristic, the preferred practical operating point is the 0.01 s LSTM, which reaches $98.747 \pm 0.022\%$ validation AUPRC with a total decision latency of 10.17 ms. Under this heuristic, it is the most attractive low-latency operating point rather than the strongest model in every comparison.

Transfer remains imperfect. The main failures are missed unstable futures rather than false alarms, and the validation-selected decision threshold transfers conservatively to unseen objects. The final chapter therefore returns to the central question and states more clearly what the results do not yet show.

5 CONCLUSION

5.1 ANSWER TO THE CENTRAL QUESTION

This dissertation asked whether short histories of sensor-invariant contact information can provide an early warning of near-future grasp instability. In simulation, the answer is yes. Across unseen test objects, all three model families learned useful predictive structure from fingertip normal and shear force histories alone. Anticipatory grasp assessment therefore does not necessarily require raw sensor-specific tactile imagery. The central hypothesis is supported: compact contact features can separate windows with stable near futures from windows with unstable near futures.

The results also give a more specific answer about model choice. On the baseline 0.20 s benchmark task, the GRU gave the strongest overall ranking performance and the most stable behaviour across random seeds. For practical use, very short windows become more attractive. Under the latency-aware validation heuristic used in section 4.5, the preferred operating point is the LSTM with a 0.01 s observation window. The 0.01 s LSTM recommendation is best read as a low-latency operating recommendation for the present heuristic, rather than as proof that the LSTM is the strongest model in every comparison.

5.2 WHAT THE RESULTS SUPPORT

The strongest claim supported by the dissertation is that force histories can support predictive transfer to unseen objects under the present simulation protocol. The pipeline combines grasp generation, replay, contact-feature extraction, temporal window construction, and evaluation on unseen objects. Within that setup, the models achieve strong ranking performance without using raw tactile images tied to one sensor design.

The friction-cone comparison in section 4.7 helps interpret this result physically. A proxy built from the ratio of tangential to normal load already performs strongly, which suggests that much of the predictive signal is consistent with the contact mechanics reviewed earlier. The learned models still perform better, indicating that short force histories contain useful temporal structure beyond that one ratio.

The results also support a narrower point about model choice. At the baseline 0.20 s setting, the CNN, GRU, and LSTM all remain close in performance. The GRU is the strongest benchmark model, but the gap is small. The small gap suggests that much of the useful signal already lies in short recent temporal patterns rather than in long temporal memory.

More broadly, the results show that a compact, sensor-invariant contact representation can work well in simulation. Such a representation may also scale more easily than collecting very large real-world datasets of raw tactile imagery.

5.3 WHAT THEY DO NOT YET SHOW

The present results still need careful interpretation. Most importantly, the study is entirely simulation-based. The learned models operate on clean simulator-derived contact features, not on tactile measurements affected by sensor noise, calibration drift, latency, lighting variation, or actuation error. The results should therefore be read as a controlled demonstration of feasibility, not as a full account of physical tactile grasping.

The underlying physics is also simplified. The replay environment treats the hand and object as rigid bodies and does not model the richer contact behaviour associated with soft or deformable materials. Surface roughness, local friction variation, materials that resist motion differently

depending on force direction, and other fine-grained contact effects are simplified. These factors matter because, in practice, whether a disturbance develops into slip often depends on more than the bulk normal and shear forces alone.

The task is also deliberately narrow. The experiments study whether a formed single-hand grasp remains stable under replay with gravity, rather than whether a robot can use the prediction online inside a closed-loop manipulation policy. The current setup therefore excludes active regrasping, external disturbances, object reorientation, and coordinated bimanual grasps. This restricted setup is useful for isolating the predictive question, but it limits how far the conclusions can be extended to richer manipulation tasks.

The evaluation is also limited in breadth. The unseen-object split is stronger than replay-level separation alone, but the object set is still small. Transfer is assessed on only two unseen test objects. All results also depend on one hand morphology and one labelling design.

The benchmark is also window-level rather than grasp-level. A single grasp can contribute several sampled windows, and each of those windows is scored separately. The present results therefore show how well the model ranks or classifies sampled windows within a grasp. They do not yet show how reliably one grasp episode receives one timely and useful warning.

For these reasons, the dissertation can claim predictive transfer under the present protocol, but not universal robustness across object families, materials, task conditions, or grasp-level warning objectives.

5.4 FUTURE WORK

Future work can reuse most of the current pipeline. The predictive framing introduced in sections 3.3.3 and 3.3.4 does not depend on this exact rigid-body study, and the same training loop and unseen-object evaluation structure can be retained while the setting becomes more realistic.

The most immediate next step is to broaden the evaluation. A larger and more varied object set, with greater diversity in shape, mass, frictional behaviour, and surface texture, would make the generalisation claim much stronger. A broader evaluation should be paired with grasp-level warning analyses as well as the present window-level benchmark. That would test not only how well the model scores sampled windows within a grasp, but also whether it gives one warning early enough to be useful for the grasp as a whole.

A second priority is to make the contact model more realistic. Extending the simulator to include compliant fingertips, soft objects, or other deformable-body effects would allow the work to move beyond rigid-body grasping. A more realistic contact model would also test whether the same predictive structure survives when contact patches can deform and redistribute over time. In that setting, the current force signals may no longer be sufficient on their own if deformation, contact area, and local stress distribution become central to the onset of instability.

Future work should also revisit the input representation. The present force aggregation is intentionally compact and sensor-invariant, but richer features that keep more spatial structure may help with the remaining false negatives near transition boundaries. A natural extension would keep the same labels and evaluation protocol. Such an extension would replace the feature-extraction stage with contact representations that preserve local force distribution, contact location, or synthetic tactile image structure.

Another important step is to place the prediction model inside a live controller. The current experiments show that instability can often be anticipated, but they do not show that a controller can use those predictions to prevent failure. In that setting, the model would guide grip or posture changes during execution instead of scoring replay windows after the fact. The latency analysis in section 4.5 would still matter there.

Finally, some of the most important next evaluations are sim-to-real transfer, explicit analysis of decision-threshold transfer on unseen objects, richer manipulation settings such as external disturbances and in-hand reorientation, and coordinated two-hand grasps. These directions would help show which parts of the present result reflect a general predictive principle and which depend on the present protocol.

BIBLIOGRAPHY

[1] C. Piazza, G. Grioli, M. Catalano and A. Bicchi. 'A Century of Robotic Hands'. en. In: *Annual Review of Control, Robotics, and Autonomous Systems* 2.1 (May 2019), pp. 1–32. DOI: 10.1146/annurev-control-060117-105003.

[2] K. Friston et al. 'Active inference and learning'. en. In: *Neuroscience & Biobehavioral Reviews* 68 (Sept. 2016), pp. 862–879. DOI: 10.1016/j.neubiorev.2016.06.022.

[3] G. Pezzulo, F. Rigoli and K. J. Friston. 'Hierarchical Active Inference: A Theory of Motivated Control'. In: *Trends in Cognitive Sciences* 22.4 (Apr. 2018), pp. 294–306. DOI: 10.1016/j.tics.2018.01.009.

[4] A. J. Nagengast, D. A. Braun and D. M. Wolpert. 'Risk-Sensitive Optimal Feedback Control Accounts for Sensorimotor Behavior under Uncertainty'. en. In: *PLoS Computational Biology* 6.7 (July 2010). Ed. by J. Diedrichsen, e1000857. DOI: 10.1371/journal.pcbi.1000857.

[5] R. S. Johansson and J. R. Flanagan. 'Coding and use of tactile signals from the fingertips in object manipulation tasks'. en. In: *Nature Reviews Neuroscience* 10.5 (May 2009), pp. 345–359. DOI: 10.1038/nrn2621.

[6] W. Yuan, S. Dong and E. Adelson. 'GelSight: High-Resolution Robot Tactile Sensors for Estimating Geometry and Force'. en. In: *Sensors* 17.12 (Nov. 2017), p. 2762. DOI: 10.3390/s17122762.

[7] M. Lambeta et al. 'DIGIT: A Novel Design for a Low-Cost Compact High-Resolution Tactile Sensor with Application to In-Hand Manipulation'. en. In: *IEEE Robotics and Automation Letters* 5.3 (July 2020), pp. 3838–3845. DOI: 10.1109/LRA.2020.2977257.

[8] B. Ward-Cherrier et al. 'The TacTip Family: Soft Optical Tactile Sensors with 3D-Printed Biomimetic Morphologies'. In: *Soft Robotics* 5.2 (Apr. 2018), pp. 216–227. DOI: 10.1089/soro.2017.0052.

[9] A. Yamaguchi and C. G. Atkeson. 'Implementing tactile behaviors using FingerVision'. In: *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*. Nov. 2017, pp. 241–248. DOI: 10.1109/HUMANOIDS.2017.8246881.

[10] Z. He et al. 'TacMMs: Tactile Mobile Manipulators for Warehouse Automation'. In: *IEEE Robotics and Automation Letters* 8.8 (Aug. 2023), pp. 4729–4736. DOI: 10.1109/LRA.2023.3287363.

[11] A. Daniels et al. 'Grasping in Uncertain Environments: A Case Study For Industrial Robotic Recycling'. In: *2023 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. Oct. 2023, pp. 3514–3521. DOI: 10.1109/SMC53992.2023.10394008.

[12] W. Chen et al. 'Tactile Sensors for Friction Estimation and Incipient Slip Detection—Toward Dexterous Robotic Manipulation: A Review'. In: *IEEE Sensors Journal* 18.22 (Nov. 2018), pp. 9049–9064. DOI: 10.1109/JSEN.2018.2868340.

[13] Y. Lu et al. 'A Neuromorphic Incipient Slip Detection System using Papillae Morphology'. In: *IEEE Robotics and Automation Letters* 11.3 (Mar. 2026), pp. 2802–2809. DOI: 10.1109/LRA.2026.3655298.

[14] Q. K. Luu et al. *ManiFeel: Benchmarking and Understanding Visuotactile Manipulation Policy Learning*. May 2025. DOI: 10.48550/arXiv.2505.18472.

[15] Q. Liu et al. 'VTDexManip: A Dataset and Benchmark for Visual-Tactile Pretraining and Dexterous Manipulation with Reinforcement Learning'. en. In: *International Conference on Learning Representations*. Ed. by Y. Yue et al. Vol. 2025. 2025, pp. 90582–90607.

[16] H. Li et al. 'Classification of Vision-Based Tactile Sensors: A Review'. en. In: *IEEE Sensors Journal* 25.19 (Oct. 2025), pp. 35672–35686. DOI: 10.1109/JSEN.2025.3599236.

[17] S. Suresh et al. 'NeuralFeels with neural fields: Visuotactile perception for in-hand manipulation'. en. In: *Science Robotics* 9.96 (Nov. 2024), eadl0628. DOI: 10.1126/scirobotics.adl0628.

[18] M. Comi et al. 'TouchSDF: A DeepSDF Approach for 3D Shape Reconstruction using Vision-Based Tactile Sensing'. In: *IEEE Robotics and Automation Letters* 9.6 (June 2024), pp. 5719–5726. DOI: 10.1109/LRA.2024.3396054.

[19] M. Yang et al. 'AnyRotate: Gravity-Invariant In-Hand Object Rotation with Sim-to-Real Touch'. In: *Proceedings of the 8th Conference on Robot Learning*. Ed. by P. Agrawal, O. Kroemer and W. Burgard. Vol. 270. Proceedings of Machine Learning Research. PMLR, Nov. 2025, pp. 4727–4747.

[20] H. Li et al. 'BioTacTip: A Soft Biomimetic Optical Tactile Sensor for Efficient 3D Contact Localization and 3D Force Estimation'. en. In: *IEEE Robotics and Automation Letters* 9.6 (June 2024), pp. 5314–5321. DOI: 10.1109/LRA.2024.3387111.

[21] S. Carpin, S. Liu, J. Falco and K. Van Wyk. *Multi-Fingered Robotic Grasping: A Primer*. July 2016. DOI: 10.48550/arXiv.1607.06620.

[22] J. R. Taylor, E. M. Drumwright and J. Hsu. 'Analysis of grasping failures in multi-rigid body simulations'. en. In: *2016 IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*. San Francisco, CA, USA: IEEE, Dec. 2016, pp. 295–301. DOI: 10.1109/SIMPAR.2016.7862410.

[23] S. Dong, W. Yuan and E. H. Adelson. 'Improved GelSight Tactile Sensor for Measuring Geometry and Slip'. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Sept. 2017, pp. 137–144. DOI: 10.1109/IROS.2017.8202149.

[24] S. Dong, D. Ma, E. Donlon and A. Rodriguez. 'Maintaining Grasps within Slipping Bounds by Monitoring Incipient Slip'. In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, May 2019, pp. 3818–3824. DOI: 10.1109/ICRA.2019.8793538.

[25] C. J. Ford et al. 'Shear-Based Grasp Control for Multifingered Underactuated Tactile Robotic Hands'. In: *IEEE Transactions on Robotics* 41 (2025), pp. 3113–3128. DOI: 10.1109/TRO.2025.3563046.

[26] N. Jawale et al. 'Learned Slip-Detection-Severity Framework using Tactile Deformation Field Feedback for Robotic Manipulation'. In: *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Oct. 2024, pp. 13569–13576. DOI: 10.1109/IROS58592.2024.10802687.

[27] T. Ayral, S. Aloui and M. Grossard. *Reactive Slip Control in Multifingered Grasping: Hybrid Tactile Sensing and Internal-Force Optimization*. Feb. 2026. DOI: 10.48550/arXiv.2602.16127.

[28] R. Calandra et al. 'The Feeling of Success: Does Touch Sensing Help Predict Grasp Outcomes?' In: *Proceedings of the 1st Annual Conference on Robot Learning*. Ed. by S. Levine, V. Vanhoucke and K. Goldberg. Vol. 78. Proceedings of Machine Learning Research. PMLR, Nov. 2017, pp. 314–323.

[29] B. S. Zapata-Impata, P. Gil and F. Torres. 'Tactile-Driven Grasp Stability and Slip Prediction'. en. In: *Robotics* 8.4 (Sept. 2019), p. 85. DOI: 10.3390/robotics8040085.

[30] E. Todorov, T. Erez and Y. Tassa. 'MuJoCo: A physics engine for model-based control'. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2012, pp. 5026–5033. DOI: 10.1109/IROS.2012.6386109.

[31] M. Mittal et al. 'Isaac Lab: A GPU-Accelerated Simulation Framework for Multi-Modal Robot Learning'. In: *arXiv preprint arXiv:2511.04831* (2025).

[32] S. Wang, M. Lambeta, P.-W. Chou and R. Calandra. 'TACTO: A Fast, Flexible, and Open-Source Simulator for High-Resolution Vision-Based Tactile Sensors'. In: *IEEE Robotics and Automation Letters* 7.2 (Apr. 2022), pp. 3930–3937. DOI: 10.1109/LRA.2022.3146945.

[33] Y. Lin, J. Lloyd, A. Church and N. F. Lepora. 'Tactile Gym 2.0: Sim-to-Real Deep Reinforcement Learning for Comparing Low-Cost High-Resolution Robot Touch'. en. In: *IEEE Robotics and Automation Letters* 7.4 (Oct. 2022), pp. 10754–10761. DOI: 10.1109/LRA.2022.3195195.

[34] E. Su et al. 'Sim2Real Manipulation on Unknown Objects with Tactile-based Reinforcement Learning'. In: *2024 IEEE International Conference on Robotics and Automation (ICRA)*. May 2024, pp. 9234–9241. DOI: 10.1109/ICRA57147.2024.10611113.

[35] S. Luo et al. 'Tactile Robotics: An Outlook'. In: *IEEE Transactions on Robotics* 41 (2025), pp. 5564–5583. DOI: 10.1109/TRO.2025.3608686.

[36] T. Liu et al. 'Synthesizing Diverse and Physically Stable Grasps with Arbitrary Hand Structures using Differentiable Force Closure Estimator'. In: *IEEE Robotics and Automation Letters* 7.1 (Jan. 2022), pp. 470–477. DOI: 10.1109/LRA.2021.3129138.

[37] R. Wang et al. 'DexGraspNet: A Large-Scale Robotic Dexterous Grasp Dataset for General Objects Based on Simulation'. In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2023, pp. 11359–11366. DOI: 10.1109/ICRA48891.2023.10160982.

[38] C. C. Christoph et al. 'ORCA: An Open-Source, Reliable, Cost-Effective, Anthropomorphic Robotic Hand for Uninterrupted Dexterous Task Learning'. In: *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Oct. 2025, pp. 8503–8510. DOI: 10.1109/IROS60139.2025.11246681.

[39] *Shadow Dexterous Hand Series - Research and Development Tool*. en. Sept. 2023. URL: https://shadowrobot.com/dexterous-hand-series/ (visited on 16/03/2026).

[40] J. Romero, D. Tzionas and M. J. Black. 'Embodied Hands: Modeling and Capturing Hands and Bodies Together'. In: *ACM Transactions on Graphics* 36.6 (Nov. 2017), pp. 1–17. DOI: 10.1145/3130800.3130883.

[41] *Allegro Hand*. URL: https://www.allegrohand.com/sub/product/p.php?idx=1 (visited on 16/03/2026).

[42] B. Calli et al. 'The YCB object and Model set: Towards common benchmarks for manipulation research'. In: *2015 International Conference on Advanced Robotics (ICAR)*. July 2015, pp. 510–517. DOI: 10.1109/ICAR.2015.7251504.

[43] S. K. Srinivas, Y. Shukla, A. Arnold and S. Chitta. 'GraspFactory: A Large Object-Centric Grasping Dataset'. In: *CoRL 2025 Workshop on Robot Data*. 2025.

[44] L. Downs et al. 'Google Scanned Objects: A High-Quality Dataset of 3D Scanned Household Items'. In: *2022 International Conference on Robotics and Automation (ICRA)*. May 2022, pp. 2553–2560. DOI: 10.1109/ICRA46639.2022.9811809.

[45] P. J. Rousseeuw and C. Croux. 'Alternatives to the Median Absolute Deviation'. en. In: *Journal of the American Statistical Association* 88.424 (Dec. 1993), pp. 1273–1283. DOI: 10.1080/01621459.1993.10476408.

[46] C. Hager-Ross, K. Cole and R. Johansson. 'Grip-force responses to unanticipated object loading: load direction reveals body- and gravity-referenced intrinsic task variables'. en. In: *Experimental Brain Research* 110.1 (June 1996). DOI: 10.1007/BF00241383.

DEPARTMENT OF ENGINEERING MATHEMATICS
UNIVERSITY OF BRISTOL
BEACON HOUSE, QUEENS ROAD, BS8 1QU
TEL: +44 117 928 9000
BRISTOL.AC.UK