# T0.5 Validation Protocol for Discrete-State Hypothesis and Transfer Validity

## Introduction and Scope

T0.5 focuses on **validation** of two critical components from the T0.1–T0.4 development stages: (1) the **Discrete-State Hypothesis** (i.e., using a finite set of codebook states to represent behaviors) and (2) the **Behavioral Alignment Framework** (mapping human motions to quadruped behaviors). This validation stage will rigorously test whether our learned Vector Quantization (VQ) / Finite State Quantization (FSQ) codes truly capture meaningful behavioral states and whether the intent-based behavior mapping holds up under real-world conditions. The goal is to ensure that the representations and mappings developed so far are not just theoretical constructs but have practical validity and reliability.

We will design comprehensive experiments to address both aspects, prioritizing the discrete-state representation first (as it underpins everything else) and the intent-mapping second. **Key questions include:** Do the learned discrete codes correspond to actual, distinct behaviors (or are they arbitrary)? Is our choice of codebook size and levels justified beyond a theoretical rate-distortion argument? Can human activity data reliably predict quadruped behavior intents, and are those predictions well-calibrated and trustworthy? We also need to identify any gaps – notably the lack of real quadruped sensor data – and plan how to mitigate them.

This document lays out a **detailed T0.5 validation protocol** including the experimental design, statistical analysis plan, data collection approach, and success criteria (with clear go/no-go decisions). Following the guidelines from previous phases, all results will be accompanied by uncertainty estimates (e.g. confidence intervals) and we will avoid any validation claims we cannot support (for example, we explicitly **will not** attempt kinematic/joint-angle transfer validation in T0.5, per the pivot in T0.4).

## Primary Validation Focus Areas

Our validation efforts are split into two main focus areas, each with its own priorities and criteria:

### A. **Discrete-State Hypothesis Validation** (Priority 1)

**Core Question:** *Do FSQ/VQ codes capture meaningful behavioral states in the data?*  
This is the highest priority because if the discrete quantized states are not meaningful, the entire representation framework could be flawed. We will validate the **structure and usage of the codebook** obtained from our VQ model (the one defined by rate-distortion optimization). Key points to examine:

* **Optimal Codebook Levels:** Our model currently uses a hierarchical codebook with levels [7, 6, 5, 5, 4] (as determined by rate-distortion theory) compared to an earlier empirical choice of [8, 6, 5, 5, 4]. We need to verify that using 7 codes at the first level (instead of 8) is truly beneficial and not a post-hoc rationalization. Does this change improve performance or interpretability? This will involve an **ablation study** to compare the two configurations.
* **Code Utilization:** Preliminary results showed only ~7.4% of the 4800 possible VQ codes are being utilized. Such low usage could indicate over-parameterization (too large a codebook) or a distributional skew (many codes never get assigned). We must investigate why utilization is low – e.g., are most behaviors clumping into a small subset of codes? – and whether this harms the representation. A well-designed codebook should utilize a healthy fraction of its codes if it’s capturing diverse behaviors.
* **Behavioral Separability:** If the discrete states are meaningful, data points assigned to different codes should correspond to distinct behaviors. We will test **cluster separability** in the quantized latent space (for instance, do all “walking” instances fall into a few specific codes separate from “rest” or “trotting” codes?). Ideally, the VQ codes partition the data by behavior class better than chance.
* **Temporal Consistency:** Real behaviors have temporal continuity – e.g., if the animal is trotting, it should stay in that state for a contiguous period, not flip arbitrarily from frame to frame. We will validate that the sequence of assigned codes over time is temporally smooth and consistent with actual behavior durations. Sudden frequent jumps between states might indicate the code is capturing noise or uninterpretable features rather than stable behaviors.

By confirming the above, we establish that the discrete representation (FSQ/VQ codebook) is an appropriate and valid abstraction of the quadruped’s behavior space.

### B. **Behavioral Alignment Framework Validation** (Priority 2)

**Core Question:** *Does the intent-based mapping from human data to quadruped behaviors produce valid and useful results?*  
This is the second priority – assuming the discrete states are meaningful, we then ask if our method of determining *which* behavior (state) to trigger based on human movement intent is sound. Key validation items:

* **Intent Classification Accuracy:** Using human IMU data (from wearable sensors in the PAMAP2 dataset) we derive an “intent” or activity category that is then mapped to a corresponding quadrupedal behavior. We need to measure how accurately we can classify these human intents. For example, can we distinguish when the human is signaling “rest” versus “walk” or “run (trot)” reliably? This will likely involve supervised learning evaluation on the PAMAP2 data labeled with our four quadruped intent categories (e.g., mapping specific human activities to {Rest, Walk, Trot, Transition}). High classification accuracy is needed to trust the intent mapping pipeline.
* **Behavior Library Matching:** We assume that each human activity (or intent) corresponds to a quadruped behavior in our library (e.g., human jogging intent maps to the robot trotting gait). We should verify the validity of these mappings. This could involve expert judgment or literature (ensuring, for instance, that the chosen quadruped gait for each human activity is appropriate and covers the needed spectrum of behaviors). Essentially, we validate that the “behavior library” we picked for the robot is being correctly selected by the human intent classification.
* **Uncertainty Calibration:** For any predictions the system makes (especially intent classification), we will evaluate the confidence or uncertainty estimates. A well-calibrated model should only be confident when it’s likely correct. We target an **Expected Calibration Error (ECE)** of < 0.1 (i.e., less than 10% discrepancy between predicted probability and actual outcomes). Proper calibration means if the system says it’s 90% confident in “trot,” it should be correct about 90% of the time for those predictions. This is crucial for safety – if the model is unsure, it can signal for human intervention or choose a safe default rather than act on a guess.
* **No Direct Kinematic Transfer Validation:** As decided in the T0.4 stage pivot, we will **not** attempt to validate kinematic equivalence (i.e., we are not checking if human joint movements exactly map to quadruped leg angles – that is outside our current scope). This keeps our focus on high-level behavior intent, not low-level motion matching.

By validating these aspects, we ensure the pipeline from **human movement → intent detection → discrete behavior code → robot action** is sound, at least in terms of categorization and decision logic (if not exact kinematics).

## Available Data Sets and Proposed Data Strategy

To perform these validation experiments, we must leverage existing datasets and generate new data where needed. Below are the data sources we will use, along with strategies to address current gaps (especially the lack of real quadruped sensor data):

* **Human IMU Dataset (Primary):** We will use **PAMAP2**, a public dataset of human physical activities collected via wearable IMU sensors. We have already processed 27,281 time-window samples from PAMAP2, mapping the human motions into our quadruped’s format (i.e., extracting features relevant to quadruped gait where possible). PAMAP2 provides ground truth labels for activities like walking, jogging, sitting, etc., which we will map to our defined intent categories (for example, PAMAP2 “lying” and “sitting” -> quadruped “Rest”; “walking” -> “Walk”; “running/jogging” -> “Trot”; transitional movements or undefined -> “Transition”). This dataset is central for validating intent classification and indirectly validating the discrete states (through cluster analysis by activity).
* **Synthetic Dataset (Generated):** To validate the FSQ discrete states with known ground truth, we will create synthetic sensor data sequences where the underlying “true” state is known at each time. For example, we can simulate IMU readings or other proxy features for a sequence of behaviors like rest -> walk -> trot transitions with controlled variations. Because we know the true state at each moment in the synthetic data, we can test whether our VQ model correctly identifies and segments those states. Synthetic data will be especially useful for **Tier 1** validation (detailed below) since it provides a ground truth discrete sequence. We might use simple physics models or procedural generation to simulate accelerometer patterns for different gaits, ensuring they are realistic enough to challenge the model but controlled enough for evaluation.
* **Reference Quadruped Dataset (Real Quadruped Data – *Needed*):** A critical gap is the absence of real quadruped sensor data in our current pipeline. For T0.5, we aim to gather at least a limited set of real quadruped data to test transfer validity. Ideally, we want on-body IMU or motion capture data from a dog (or a robot in quadruped mode) performing basic behaviors: resting, walking, trotting, and perhaps playful movements. Options to obtain this include:
* Searching for **public datasets** – e.g., any research dataset of dog gait IMUs or quadruped robot sensors. (Possibilities: the TUM library for quadruped robotics, or datasets from Stanford’s dog biomechanics studies). If something like a “Stanford Dogs” dataset exists for motion, that could serve as a reference.
* **Collaboration** – reaching out to a veterinary school or animal motion research lab to get sample inertial data from dogs. Even a small number of samples (on the order of 100–1000 data windows) would be extremely valuable.
* **Synthetic from biomechanical simulators** – using a physics engine or existing quadruped model (like in PyBullet or MuJoCo) to generate IMU data from simulated dog motions. This is a fallback if real data is unattainable, though we must be cautious as simulator data might not capture all real-world nuances.
* **Video-derived data** – as an unconventional approach, we could use smartphone videos of dogs and apply pose estimation to approximate IMU readings or at least footfall timing. This is less direct but could provide some validation of gait timing and transitions.

Our goal for real quadruped data is modest: we need enough samples to test if the behaviors and states recognized from human data make sense on an actual quadruped. We do not require a huge dataset at T0.5, but some real data is important to avoid solely relying on human->robot analogies.

* **Augmented Datasets (Robustness Tests):** We will also create augmented versions of the human data to test robustness. For instance, we can take PAMAP2 sequences and add noise or perturbation (e.g., slight time warping to simulate phase shifts in gait cycles, or adding drift to simulate sensor calibration issues). The aim is to ensure our system is not overly brittle – small variations in input shouldn’t completely change the output state. An **augmented PAMAP2 + phase perturbations** dataset will help validate that the discrete states and classification remain stable under minor fluctuations, reflecting real-world conditions where sensor data is never perfectly clean.

By combining these datasets – **primary human data, synthetic controlled data, reference quadruped data, and augmented data** – we cover a spectrum of scenarios for validation. Each serves a purpose: human data to anchor our intent mapping, synthetic for ground truth state checks, real quadruped to verify transfer assumptions, and augmented for noise-robustness testing.

## Validation Targets and Approach (Three Tiers)

To systematically cover the validation objectives, we propose a **three-tiered approach** to validation, from tightly controlled tests to real-world checks:

* **Tier 1: Synthetic Ground-Truth Validation** – *“Does the FSQ model recover known states?”*  
  In this tier, we use the **synthetic dataset** where we fully know the underlying state sequence. For example, we might script a sequence like 5 seconds of “rest” data, 10 seconds of “walk” data, 10 seconds of “trot” data, etc. Using these sequences, we validate that the FSQ/VQ encoding can correctly identify and segment those distinct states. We will measure **state recovery accuracy** (what fraction of time does the VQ state match the true state, allowing some tolerance around transition boundaries) and confusion (do any states get mistaken for another consistently?). This is a direct test of the discrete-state hypothesis under ideal conditions. If our model cannot recover states when the data is “clean” and well-separated, it likely won’t work on messier real data – hence this is a fundamental check.
* **Tier 2: Expert-Annotated Validation (Human Activity Labels)** – *“Do learned states align with human-labeled activities?”*  
  Here we leverage the fact that PAMAP2 comes with **activity annotations** from human subjects (e.g., labels like Sitting, Walking, Running, Cycling, etc.). We will map these human activities into our quadruped behavior categories as best as possible (noting that some human activities might not have a direct quadruped equivalent but we’ll focus on those that do). Using those labels as a proxy for “ground truth behavior,” we evaluate how the **discrete FSQ codes** cluster or distribute with respect to the labeled activities. For instance, we expect all data labeled “walking” should mostly belong to one or a small subset of VQ codes corresponding to the walking gait behavior. We will use **clustering validation metrics** such as:
* *Adjusted Mutual Information (AMI):* measuring how much agreement there is between clusters (VQ codes) and the labeled categories, adjusted for chance. High AMI would indicate the code assignments are not random with respect to actual activities.
* *Silhouette Score:* measuring how well-separated the data are when grouped by VQ codes (compared to their nearest alternate cluster). This tells us if the model formed tight, distinct clusters of data in feature space, which we expect if codes correspond to distinct behaviors.
* *Temporal consistency within labeled segments:* For each contiguous segment of a known activity (say 60 seconds of a person walking), we check how often the VQ code stays consistent or how many times it transitions. Ideally, within a single activity segment, the model should not constantly hop between different discrete states – maybe one or two state changes at most if it’s capturing phases, but not jittering every second. This complements silhouette score by adding the time dimension: correct behavior identification should persist over time until the activity changes.

This tier validates that, *in real human data with labels*, our discrete states are meaningful and correspond to intuitive behaviors. A strong result here is if each activity corresponds clearly to one or two codes and rarely overlaps with others except during true transitions.

* **Tier 3: Behavioral Event Alignment (Signal-Level Markers)** – *“Do state transitions align with known gait events or phase changes?”*  
  The final tier goes into a more granular check using signal-level **behavioral markers**. Even without explicit labels, certain features in the sensor data indicate events (especially in periodic motions like gait):
* For instance, **footfall or gait cycle events** can sometimes be inferred from accelerometer peaks (e.g., a periodic spike each time a foot hits the ground in walking).
* **Phase transitions** in periodic motion might be seen in gyroscope or accelerometer patterns (e.g., a shift from steady walking rhythm to a pause or a faster rhythm if transitioning to running).

We will identify such markers in the data (both human and any quadruped data we get). The validation here is: when these known events or transitions occur, does the **discrete state** assigned by the model correspondingly change (or reflect that event)? For example, if a series of footfall peaks suddenly double in frequency (indicating a person went from walking to running), we should see the VQ state change from “walk” to “trot/run” around the same time. Or, if a person stops moving (no more footfall impacts), we should see a state change to “rest/idle.” We will validate the **temporal alignment** between such physical events and our model’s state transitions. This ensures the model isn’t assigning states at arbitrary times, but rather in sync with real behavioral change.

Another aspect is to verify that during steady-state motion (constant gait), the model’s state remains stable and doesn’t flicker erratically in absence of external change. This tier thus checks the *fidelity* of the state sequence against known physical cues in the raw data.

By structuring validation in these three tiers (synthetic controlled tests, real labeled data comparison, and signal-level alignment), we gain confidence at multiple levels of granularity. Each tier builds on the previous: if Tier 1 passes but Tier 2 fails, perhaps the model can distinguish obvious states but not subtler real ones (indicating a need for refinement). If Tier 2 passes but Tier 3 fails, maybe the model clusters align with labels on average but the timing of transitions is off (indicating potential lag or sensitivity issues). We will use these results to diagnose any shortcomings precisely.

## Specific Validation Questions and Challenges

From the T0.4 review and earlier discussions, several **specific questions** were raised that our T0.5 experiments must address explicitly. We list them here along with how the validation will tackle each:

1. **FSQ Codebook Levels – Validity of [7,6,5,5,4]:**  
   *Question:* Is our chosen codebook architecture with levels [7,6,5,5,4] truly optimal for capturing behaviors, or is it just a post-hoc justification? Would [8,6,5,5,4] (the earlier empirical choice) have been just as good or better?  
   *Plan:* In Experiment E1 (detailed below), we will perform an **ablation/comparison study**. We’ll train or evaluate the system with both configurations and compare metrics like reconstruction error (for VQ fidelity) and behavior classification accuracy. We will also compare how many codes each scheme actually uses and how well-separated the behaviors are. If [7,6,5,5,4] is valid, we expect it to perform comparably or superior to [8,6,5,5,4] on these metrics, potentially with benefits like less overfitting or better code utilization. If it underperforms, that might mean our rate-distortion analysis was off or over-optimized in theory without practical benefit. This addresses any skepticism that the chosen number “7” (instead of 8) at the top level is arbitrary – we will have empirical evidence to back the choice.
2. **Code Utilization – Why Only 7.4% Usage?**  
   *Question:* We found that of the total codebook (around 4800 possible code combinations across all levels), only ~7.4% are actually seen in the data. Is this a problem? Does it indicate wasted capacity or a bad distribution (e.g., codes unused because data is imbalanced)?  
   *Plan:* We will analyze code usage statistics in detail. This includes looking at the frequency of each top-level code and deeper code combinations. Possibly, many codes were allocated for behaviors or variations that did not occur in our dataset. We will test if reducing the codebook (like going from 8 to 7 top-level codes) increases utilization percentage (which would be positive). We will also see if the usage is concentrated: e.g., maybe 1 or 2 codes account for 50% of data – that would be a red flag that the model isn’t differentiating much. Part of the solution might be to generate **more diverse synthetic data** to force use of more codes, or ensure the clustering algorithm wasn’t too conservative. If code utilization remains very low, we might consider pruning the codebook in future iterations. However, it’s not inherently bad to have some slack (unused codes) if those codes are meant for edge cases not seen yet – this could be a distribution issue. The validation will clarify whether the low utilization is due to model design flaws or simply lack of diverse input; if the former, that might be a no-go signal for the approach unless fixed.
3. **Behavioral Coherence – Do Discrete States Correspond to Real Behaviors?**  
   *Question:* Are the discrete states we get out actually *coherent behaviors* that make sense (e.g., a state corresponds to “trotting gait” or “idle” or “transitional movement”)? Or are they unintelligible mixtures? Essentially, is our assumption of discrete behavior states valid for the data?  
   *Plan:* This is addressed in Experiments E2 and E3. We will use clustering analyses (Exp. E2) to see if states align with known categories of activity. Additionally, by examining some of the **reconstructed or prototypical signals for each state** (if our VQ allows reconstruction, we can inspect what a given code’s typical output looks like), we can have experts label them. For example, if one code corresponds to a signal pattern of steady periodic motion (likely a gait) versus another code that corresponds to flat/inactive signals (likely rest), that’s a good sign. We might also visualize t-SNE or PCA of the latent embeddings colored by VQ code to see if they form sensible clusters. If we find that what we thought was a “discrete state” is actually mixing two very different behaviors (e.g., half the time code X appears during walking, half during jogging), then the discrete assumption may need refinement (maybe we need more states or a different encoding). On the other hand, clear coherence (one code = one behavior type) would validate our approach. We expect some states to correspond to transitional or mixed behaviors (not all behaviors are perfectly discrete), but the main steady behaviors (rest, walk, trot) should have dedicated states.
4. **Uncertainty Calibration and Errors**:  
   *Question:* Are the model’s uncertainty estimates reliable, and does high uncertainty indeed correlate with higher prediction errors? In other words, when the model is unsure (high entropy in the intent classifier or low confidence in state assignment), does it in fact often make mistakes in those instances? And conversely, when it’s very sure (say 99% confidence), is it almost always correct? Proper uncertainty calibration is critical for deciding when the robot should trust the command or when to query for human confirmation.  
   *Plan:* This is the focus of Experiment E4. We will calculate the **entropy of the output distribution** for intent classification for each sample and see how it relates to whether that sample was classified correctly or not. We expect to see a positive correlation: the more uncertain (higher entropy), the more likely an error. We’ll also compute a **calibration curve** (plotting predicted probability vs actual accuracy for binned confidence levels) and the **Brier score** or ECE (Expected Calibration Error). Our target is ECE < 0.1, meaning the model’s probabilities are within 10% of true accuracy likelihoods. For example, if we take all predictions where the model says “80% confidence”, roughly 70–90% of those should be correct (within 10% of 80%). If we find the model is overconfident (e.g., it often says 95% but is only correct 70% of the time), that indicates miscalibration and we might need to apply techniques like temperature scaling to fix it. If the model is underconfident (rare, but could happen), that’s less dangerous but still worth adjusting. We will specifically log cases of misclassification to see if they had high entropy (hopefully yes, meaning the model knew it was unsure) or if we encounter instances of **silent failures** (wrong but with high confidence – those are the worst-case and must be minimized).

Addressing each of these questions through our experiments will directly answer the concerns raised in earlier phases and ensure our approach is on solid ground. If any question yields an unsatisfactory answer (e.g., “No, the discrete states do not align with real behaviors” or “No, the model is making confident errors without knowing”), we will mark that as a **no-go** or at least an area for serious revision before moving beyond T0.5.

## Key Constraints and Considerations

In designing and executing T0.5 validation, we must keep in mind several constraints and decisions made earlier, so that we **do not venture outside scope or break assumptions**:

* **No Kinematic Joint-Level Validation:** We will **not** attempt to validate correspondence at the level of joint kinematics or detailed motion trajectories. Our system is intended to capture *behavioral state* (like gait modes) rather than precise motion imitation. Therefore, T0.5 experiments avoid any requirement of, say, comparing a dog’s leg angle to a human’s limb angle. We acknowledge this limitation; our validation focuses on higher-level behavior categories.
* **Focus on Broad Behavioral Categories:** The validation revolves around key broad behaviors – currently identified as **Rest, Walk, Trot,** and a catch-all **Transition/Other** category (for any movement not fitting the primary three, or for transitions between them). These categories were derived from our intent framework. When mapping human data or analyzing clusters, we will use these categories as reference points. We are **not** at this stage distinguishing more granular behaviors (like “gallop” or “jump” or “sit” separately) because our initial behavior library for the quadruped was limited to the essentials. The validation will check those categories; if later the project expands the library, a similar validation process would be repeated for new behaviors.
* **Inclusion of Uncertainty in Reporting:** All results we present will include some measure of uncertainty (confidence intervals, variance, etc.). For example, if we report “classification accuracy is 92%,” it will be accompanied by something like “± 2%” (perhaps a 95% confidence interval or standard error from cross-validation). Similarly, cluster metrics or reconstruction errors will have spread/variance info if available. This was emphasized in earlier phases to ensure we don’t overstate results from possibly limited data. We’ll also use appropriate statistical tests to back any claims of improvement (e.g., if [7,6,5,5,4] codebook performs better than [8,6,5,5,4], we will test that difference for significance).
* **Need for Real Quadruped Data:** We repeatedly note that a **major validation gap is real quadruped data**. Even though it’s challenging to get, our design includes obtaining some. We must be careful to not draw grand conclusions about “transfer validity” unless we test on at least a small sample of actual quadruped scenarios. Therefore, a constraint is that **T0.5 is not fully complete without at least some validation on quadruped data** – even if it’s small-scale or qualitative. If it turns out impossible to get within the timeline, we will at minimum simulate it and explicitly state that limitation (and treat it as a risk moving forward). This is a go/no-go point: proceeding without any real-world check would be risky.
* **Ethical and Safety Checks:** Although not the primary focus of T0.5, we should note any ethical/safety considerations in data collection and testing. For example, if collecting data from dogs or other animals, ensure humane treatment and necessary approvals. From a system safety perspective, if/when we test on a real robot, ensure that an unvalidated behavior does not cause harm. These considerations, while not metrics, form constraints on how we design experiments (e.g., synthetic testing first to avoid an unvetted model controlling a robot unsafely).

Keeping these constraints in mind will help maintain clarity and integrity in our validation process. Now, we proceed to the detailed experimental design.

## Detailed Experimental Design

We structure T0.5’s validation experiments into four main experiments (E1–E4), each targeting specific hypotheses and metrics, as summarized earlier. Below is a detailed plan for each:

### Experiment E1: FSQ Codebook Validation (Ablation Study)

**Hypothesis:** The *rate-distortion optimized* codebook levels [7, 6, 5, 5, 4] capture behaviors more effectively (with less redundancy) than the earlier *empirically chosen* levels [8, 6, 5, 5, 4]. This should result in equal or better reconstruction and classification performance, and possibly better code usage, with a slightly smaller top-level codebook.

**Method:**  
- We will take our VQ/FSQ model and evaluate it under two settings: one with the codebook constrained to [7,6,5,5,4] levels and one with [8,6,5,5,4] (essentially adding one extra code at the top level). If the model was originally trained with 7, we might retrain a variant with 8 to compare fairly, or if training from scratch is too costly, adjust via a smaller finetune.  
- Using the **same dataset (PAMAP2 primary dataset)**, we will encode the data using both configurations.  
- **Metrics to collect:**  
- *Reconstruction Error:* If our VQ is part of an autoencoder, measure the mean squared error (or similar) of reconstructing sensor signals from the quantized code. Lower error means the codebook is representing the data with high fidelity. We’ll compare the error distribution between 7 vs 8 top codes. We expect no significant increase in error using 7 (if our choice was sound).  
- *Behavior Classification Accuracy:* If we have an auxiliary classifier or even using nearest-neighbor in code space to predict the labeled activity, how accurate is it for each codebook variant? Essentially, does the codebook still retain discriminative features for behaviors when compressed? We could train a simple classifier on top of the VQ codes for the 4 intent categories and see which configuration yields higher accuracy.  
- *Code Utilization:* Calculate the percentage of possible codes (across all levels combined) that actually appear in the dataset encoding. Also examine top-level code frequency distribution. We expect the 7-code model might utilize a slightly higher fraction of its smaller pool (since it was optimized via rate-distortion to allocate codes where needed). If the 8-code model shows many codes nearly or completely unused, that suggests the 8th code was unnecessary. Conversely, if the 7-code model shows one code picking up heterogeneous data that were split in the 8-code model, we might see a drop in performance.  
- **Data:** Use the **PAMAP2 dataset** for this ablation, possibly supplemented by some synthetic or augmented data to see if results hold beyond exactly the training distribution. (If time permits, we might also generate a small synthetic test to explicitly see if one extra code would split a known cluster or just remain unused.)  
- **Procedure:** For each configuration, run the encoding on all test data, compute metrics, and then compare. Use statistical tests if applicable (e.g., paired t-test on reconstruction errors per sample for 7 vs 8 codes to see if difference is significant).  
- **Expected Outcome:** Ideally, the [7,6,5,5,4] model performs **on par** with [8,6,5,5,4] in reconstruction and classification, indicating no loss of fidelity, and shows equal or **higher code utilization** (utilizing its capacity more efficiently). This would validate the rate-distortion guided choice. If instead the 7-level model shows worse accuracy or much higher error, we’ll have to investigate if reducing that one code caused underfitting of some niche behavior (and decide if the slight compression is worth it).

### Experiment E2: State–Behavioral Correspondence (Clustering Analysis)

**Hypothesis:** Discrete states discovered by the VQ model correspond to meaningful behavior clusters, i.e., the model’s code assignments have high agreement with actual activity labels and form well-separated clusters in feature space. Essentially, the clustering of data induced by the VQ codes is not random or mixed, but aligned with true behaviors.

**Method:**  
- Take the **PAMAP2 dataset with ground-truth activity labels** (and their mapping to our four categories). For each data window, we already have a VQ code (from the trained model). We will perform clustering validation by treating the VQ code as the cluster identifier for that window.  
- Compute clustering metrics:  
- *Adjusted Mutual Information (AMI):* between the set of VQ code clusters and the set of ground truth activity categories. AMI near 1.0 would mean almost perfect alignment (each VQ cluster corresponds to one activity), while near 0 means no better than random. Since our activities map into four broad categories, we might actually compute AMI at two levels: fine-grained (exact PAMAP2 activities vs codes) and coarse (our 4 categories vs codes). We expect better alignment at the coarse level due to how we designed the behaviors.  
- *Silhouette Score:* compute for the feature embeddings (possibly the pre-quantization latent vectors) labeled by their VQ code. This requires a distance metric (we can use Euclidean in latent space). A high silhouette score (close to 1) means each code’s members are tight and far from other codes’ members, indicating clear clusters. A low or negative silhouette would be concerning (overlapping clusters).  
- *Temporal Consistency:* For each known activity segment in PAMAP2 (they have time segments for each labeled activity), measure how often the VQ code changes within that segment. We can define a “consistency score” e.g., the percentage of time the most frequent code in that segment occupies, or the average duration before a state change. We expect for steady activities like walking or running, a high consistency (maybe one code dominates >80% of that segment). For transitional or composite activities, it might be lower. We will report these statistics per activity type.  
- Additionally, we will do a **qualitative review**: pick a few codes and see what activities they mostly map to. If code A appears 90% during “walking” segments and rarely elsewhere, we can label code A as a “Walk” state – success. If code B appears 30% walk, 30% run, 40% other, then it’s a mixed state – perhaps a transition state or a modeling flaw. We’ll document such interpretations.  
- **Data:** PAMAP2 primarily. We might also apply the same analysis if we get any labeled **quadruped data or simulator data** (with labels like rest, walk, trot for the dog). That would be an external check: do the code clusters learned on human data still align to the analogous behaviors in dog data? (This might be limited by data volume, but even a rough check on a few sequences would be insightful.)  
- **Expected Outcome:** We hope to see a strong correspondence – e.g., a handful of codes clearly aligning with each of the major categories. A hypothetical good result: Code1 = Rest (seen mostly in sitting/lying data), Code2 = Walk (seen in walking), Code3 = Trot (seen in running/jogging), Code4 = Transition (seen during activity changes or miscellaneous movements). AMI would be significantly above 0 (maybe in the 0.5–0.8 range for coarse categories), and silhouette scores positive and moderately high. Temporal consistency should show long stretches of same code during uniform activities. If results show muddled clusters (AMI low, each code contains a mix of many activities), that means our discrete states might not align well with human-labeled concepts – possibly the model latched onto some other signal patterns or too fine-grained distinctions. That would need investigation (e.g., maybe the model splits “walk” into two states based on speed or some detail the human label didn’t distinguish; that might not be entirely bad, but we should be aware).

### Experiment E3: Intent Classification Accuracy and Mapping Validity

**Hypothesis:** Human wearable sensor data can be reliably classified into a small set of intent categories (Rest, Walk, Trot, Transition), and these intended behaviors match the correct entries in the quadruped’s behavior library. In other words, given sensor data from a person, the system can correctly infer how the robot should behave, with high accuracy and minimal confusion between categories.

**Method:**  
- We will train a **supervised intent classifier** using the PAMAP2 dataset, where the input features are human IMU-derived features (possibly the same features feeding into our VQ, or handcrafted features like mean acceleration, variance, step frequency, etc.), and the labels are one of four intent categories {Rest, Walk, Trot, Transition}. These labels will be derived from the PAMAP2 activity labels by grouping them as discussed (e.g., lying, sitting, standing -> Rest; walking -> Walk; running (and maybe cycling, since it’s rhythmic leg motion) -> Trot equivalent; anything not fitting or transitional -> Transition). We need to define this mapping clearly before training.  
- Use cross-validation (e.g., 5-fold CV or leave-one-subject-out CV if PAMAP2 has multiple subjects) to train and evaluate the classifier. This will give us robust accuracy estimates and highlight any overfitting.  
- **Metrics:**  
- *Classification Accuracy:* Overall percentage of correct predictions on the held-out data. We’ll also break this down by class (to see if, say, “Transition” is much harder to classify than “Rest” etc.).  
- *Confusion Matrix:* to identify which classes get confused with which. For instance, do some “Trot” intents get misclassified as “Walk”? That might happen if a person is jogging slowly vs walking fast – there’s a grey zone. Understanding confusions will help refine the intent definitions or thresholds.  
- *Calibration metrics:* We will record the predicted probability for the intended class and compute the **Expected Calibration Error (ECE)** as discussed, as well as perhaps plot a reliability diagram (predicted confidence vs actual accuracy). This quantifies how well-calibrated the classifier is.  
- *Precision/Recall or F1:* In case the class distribution is imbalanced (likely Rest might have more data than Trot, etc.), overall accuracy can be misleading, so we’ll look at F1 scores per class as well, to ensure each intent is handled well.  
- **Data:** Exclusively PAMAP2 for training, possibly some synthetic or augmented data for additional testing. For example, after training on real data, we might test the classifier on some synthetic sequences to see if it still picks the correct intent (this can double-check that our mapping logic holds in a controlled scenario: e.g., a synthetic trot-like IMU signal is indeed recognized as Trot intent). If we obtain any real quadruped data, we can’t directly “classify” that with the human-trained model, but we could attempt to run the human classifier on data from someone mimicking a dog’s gait (if we have such a thing) or inversely run the quadruped data through the same features as a sanity check (though the models might not directly apply). Primarily, this experiment is about the human data.  
- **Validation of Behavior Library Mapping:** This is a bit qualitative – we will double-check that each intent category corresponds to the right robot behavior. For example, if the classifier labels something as “Trot,” we assert that the robot’s trot gait is the intended action. There isn’t a numeric metric here, but we ensure consistency with design: our four classes were chosen to align with the robot’s capabilities, so if any PAMAP2 activity falls outside these (say, cycling might not map well – if we included it as Trot analog, is that valid?), we clarify how to handle it. Possibly, we exclude or mark some human activities as not applicable to the robot if needed. Success here is simply that the taxonomy of intents covers the data and matches the robot’s repertoire, which it should by construction.  
- **Expected Outcome:** We anticipate a fairly high accuracy (perhaps ~90% or more) for distinguishing rest vs walk vs trot on PAMAP2, since these have distinct sensor signatures (rest is mostly no movement, walk is moderate periodic motion, trot/run is higher intensity periodic motion). The toughest class is likely “Transition/Other” because it’s a catch-all. We’ll consider the classifier successful if it can identify the primary three states with high precision and only confuses them during ambiguous boundary conditions. An acceptable ECE < 0.1 should be achievable with techniques like Platt scaling or isotonic regression if needed (we can calibrate the model after training). If accuracy comes out lower than expected, we might need to incorporate better features, consider temporal models, or refine how we define “Transition” (maybe break it down or give the model temporal context to detect transitions). This experiment essentially validates the front-end of our pipeline (human intent recognition) in isolation.

### Experiment E4: Uncertainty Calibration and Error Analysis

**Hypothesis:** The model’s uncertainty estimates correlate with actual performance, meaning we can trust high uncertainties as warnings and high confidences as reliable. In particular, when the model (e.g., the intent classifier from E3) outputs a high-entropy (uncertain) prediction, that instance is more likely to be an error, and overall the predicted probability scores are well-aligned with true probabilities (good calibration).

**Method:**  
- Using the predictions from **Experiment E3** (and potentially from E2 if we consider the unsupervised code assignments as a form of prediction), analyze the relationship between prediction confidence and correctness.  
- Calculate **entropy** of the probability distribution for each prediction: Entropy(p) = -Σ p\_class \* log(p\_class). For a 4-class output, entropy ranges from 0 (completely certain about one class) to log(4) (max uncertainty when all classes equal probability).  
- Bucket the predictions by confidence level (e.g., 0.0–0.1, 0.1–0.2… or by predicted probability of the top class, etc.) and compute how often the prediction in each bucket was correct. Plot a **calibration curve** (also known as a reliability diagram) to visually assess calibration. Ideally, the curve should lie near the diagonal (meaning 80% confidence predictions are correct ~80% of time, etc.).  
- Compute **ECE (Expected Calibration Error)**: essentially the weighted average of the difference between those bucket accuracy and confidence values. Our target is <0.1. We may also compute the Brier Score (the mean squared error of the probability estimate against the one-hot true distribution) as a summary of both calibration and accuracy (lower Brier is better, with 0 being perfect).  
- For the **correlation analysis**, compute the correlation (Pearson’s r or Spearman) between per-sample entropy and an error indicator (1 if wrong, 0 if correct). We expect a positive correlation (errors tend to have higher entropy). We might even do a comparison of mean entropy: calculate the average entropy for correct predictions vs the average entropy for incorrect predictions. We hope to see a clear difference (incorrect ones having higher uncertainty on average).  
- If we have access to uncertainty estimates from the VQ model (for example, if it provides some posterior or if we ensemble multiple models to get disagreement), we could do a similar analysis for state assignments. However, since VQ typically gives a single code (hard assignment), there’s no direct probability. Alternatively, we might look at the reconstruction error or distance to nearest code as a proxy uncertainty for whether a data point fits well into the codebook. High reconstruction error might indicate the input doesn’t well match any learned state (which is an uncertainty of sorts). We could check if high reconstruction error cases correspond to outliers or misclassified behavior segments. This is an auxiliary analysis beyond the classifier.  
- **Data:** Primarily the results from prior experiments. So no new data collection, just analysis. We will use all the predictions on test sets from E3’s cross-validation as the sample for calibration analysis (so possibly thousands of points). If E2 produced a pseudo-prediction (like assigning states vs actual labels), we might consider that too for curiosity.  
- **Expected Outcome:** We expect to find that our classifier is reasonably well-calibrated, perhaps with ECE around, say, 0.05–0.1 (if it’s higher initially, we will apply calibration methods to bring it down). We also expect that erroneous predictions indeed have higher entropy. If we discover some errors were made with high confidence, we will scrutinize those cases – they might represent either mislabeled data or systematic biases (for instance, if the model always confidently calls “Walk” when it’s actually a slow jog, maybe our labeling or feature set can’t distinguish those well). Those insights can be looped back to refine the system. Ultimately, a strong result here would give us the green light to trust the model’s confidence indicator in deployment: e.g., if the model says it’s only 50% sure, perhaps we don’t act or we gather more data at runtime; if it says 99%, we execute the behavior directly.

Each experiment E1–E4 thus addresses different facets of validation: E1 checks the VQ design itself, E2 links the VQ output to real behaviors, E3 tests the mapping from humans to those behaviors, and E4 ensures we have a handle on when to trust these outputs.

The experiments will be carried out in parallel where possible, but some depend on the outputs of others (E3 produces data for E4, etc.). We will document each experiment’s results in detail, including any intermediate findings.

## Data Collection Protocol

For T0.5 validation, careful preparation of data is as important as the analysis. Here we outline how we will gather and prepare each type of data for the experiments:

* **Preparation of PAMAP2 Data (Human IMU):** We have already preprocessed PAMAP2 into 27,281 windows of data formatted for our model. For clarity, we will document how these windows are defined (e.g., 5-second windows with 50% overlap? what sensor axes and features?). If not already done, we will label each window with an intent category label (Rest/Walk/Trot/Transition) based on PAMAP2’s activity annotation. This likely involves creating a mapping function: for each PAMAP2 activity label (such as “lying”, “sitting”, “walking”, “running”, “ascending stairs”, etc.), assign one of the four categories or mark as “Transition/Other” if it doesn’t fit well. We may need a small script or manual review for ambiguous cases. The output is a labeled dataset (X\_human, y\_intent) for classification use, and (X\_human, code) for clustering use (with code from the VQ model). We will also hold out a portion (or use CV) to ensure we have separate training vs validation splits for classification.
* **Generation of Synthetic Dataset:** We will design a few synthetic scenarios that mimic typical sequences of behaviors:
* **Simple periodic gait simulation:** e.g., simulate an accelerometer’s vertical axis during walking and trotting. We might use sine waves of different frequencies and amplitudes to represent foot impacts. Add some noise to make it realistic.
* **Composite behavior sequences:** string together segments like 10s of “rest” (basically flat noise), then 10s of “walk-like” signal, then 10s of “trot-like” signal, etc., maybe with smooth transitions in between to simulate acceleration/deceleration. We’ll generate multiple variations (to account for different speeds or slight changes).
* **Edge cases:** maybe a very short bout of trot between rest periods (to see if model catches a quick burst), or alternating walk/trot every few seconds (to stress-test temporal consistency).
* Each synthetic sequence comes with a ground truth label per time step by construction. We’ll run these through the FSQ encoding and see what code sequence we get out, enabling direct comparison to the known truth.
* The synthetic data doesn’t need to be huge (even a dozen well-crafted sequences could suffice), but it should be rich enough to test the model. We’ll ensure to cover each discrete state multiple times.
* If possible, simulate data that produces every code in the codebook at least once (especially if code utilization was an issue, we want to see what those rarely used codes correspond to – maybe by forcing certain patterns).
* **Acquisition of Real Quadruped Data:** This is the most uncertain part. Our plan:
* First, do a thorough search for **existing datasets**. If any are found (say a dataset of dog motions with IMUs), we will download it and extract segments of rest, walking, trotting. Barring direct IMU data, even motion capture that can be converted into acceleration might be used. The protocol would involve writing a converter script or using an existing model to get acceleration from motion capture (by differentiating positions, etc.).
* If no dataset is available, we will consider a **local data collection**. For example, if a small IMU can be strapped to a dog (safely and with owner consent), record a few minutes of various activities. Alternatively, if we have a quadruped robot (like a Unitree or Boston Dynamics Spot) available, we could script it to perform the gaits and record its internal sensors. We’ll need to prepare a data collection sheet and ensure synchronization if multiple sensors are used.
* We only need basic labeling: knowing which segment is which behavior. That can be done by noting down times or having someone start/stop recording during each behavior.
* All animal-related procedures, if any, would follow ethical guidelines (not causing distress, minimal instrumentation).
* If using a simulator, we’ll script the simulator to output IMU readings (accelerometer and gyro) while the virtual quadruped does different gaits. We need to calibrate the simulator’s output units to be comparable to our data (probably in m/s^2 for accel, deg/s for gyro, etc.).
* Once collected, we’ll preprocess this quadruped data into the same feature format as PAMAP2 so that our model (trained on human data) can be applied. This likely involves normalizing sensor axes and perhaps resampling if needed to the same frequency.
* **Augmentation of Data:** For robustness tests, we will augment some PAMAP2 windows:
* **Phase perturbation:** e.g., randomly time-shift or stretch portions of a walking sequence to simulate a person who doesn’t walk in a perfectly periodic manner. The label remains the same (still “Walk”), but it’s a bit off from what the model might expect if it was used to very regular patterns. We should see if the model still classifies it as walk and assigns the same state.
* **Add noise:** e.g., add Gaussian noise to accelerometer readings, or simulate sensor dropouts/spikes. Test the effect on classification and state assignment. The model should ideally be robust to mild noise (states shouldn’t flip with every jitter).
* **Context variation:** if possible, test the model on sequences that combine behaviors in unexpected orders (the model might have mostly seen a pattern of rest -> walk -> trot, etc. in human data; what if trot happens from a standstill abruptly?). We can cut and splice some data to create novel transitions.
* These augmented data won’t necessarily be used for training, but rather as additional test cases to probe model stability.
* **Data Management:** We will maintain a clear dataset directory structure:
* human\_data/ (with sub-folders for raw, processed windows, labels, etc.)
* synthetic\_data/ (with ground truth annotations stored alongside)
* quadruped\_data/ (real data, if obtained, plus any simulator output)
* augmented\_data/ (could be within human\_data as variants, but clearly marked)  
  Each dataset used in an experiment will be version-controlled or at least checksummed, so results can be traced to the exact data version. We will also likely integrate this into our experiment tracking system so that each run (for E1–E4) points to the specific data snapshot.

Throughout collection and preprocessing, we will document any issues (for example, if PAMAP2 had missing values, or if sensor axes needed alignment) to ensure transparency in the validation results.

## Statistical Analysis Plan

To turn the results of our experiments into solid conclusions, we will apply appropriate statistical analysis methods for each measured outcome. The plan covers how we will summarize data, what statistical tests or calculations to perform, and how to decide significance or confidence:

* **Experiment E1 (Codebook Comparison):**
* We will compute the mean and standard deviation of reconstruction errors for each codebook setting. If multiple data points are available per setting (which they are, since we have thousands of windows), we can do a **paired comparison** of errors (since each data window can be encoded with both models, assuming we apply both to the same data). A paired t-test (or non-parametric equivalent like Wilcoxon signed-rank if errors are not normally distributed) will be used to test if there is a significant difference in reconstruction error between 7-code and 8-code models.
* For classification accuracy (if we train a classifier on top of codes), that would involve training separate classifiers for each codebook scenario. We could compare accuracies with a McNemar’s test or just report them if training/test splits differ. However, since classification here might not be the primary metric, we might focus on the reconstruction and clustering differences.
* Code utilization being a single number (%) doesn’t lend itself to stat testing directly, but we will interpret it in context. If one model uses e.g. 7% and the other 5%, we note the increase. If possible, we could simulate uncertainty by bootstrap (resample windows and see how utilization varies) to give a confidence interval on that percentage (though utilization tends to be deterministic given all data).
* **Significance criteria:** We’re not setting a strict p-value threshold (since this is more about practical significance), but generally p<0.05 will be considered evidence of a real difference in errors. We will also consider effect size: e.g., if error difference is statistically significant but extremely small (negligible effect), we might still choose the simpler model.
* **Experiment E2 (Clustering metrics):**
* We will report the AMI and silhouette scores for the clustering of VQ codes vs labels. To judge these, we might compare against some baselines: e.g., AMI of 0 would be random, AMI of 1 is perfect. We can generate a baseline by random shuffling labels to see what random AMI we get (should be ~0). If possible, also compare to using raw k-means clustering on features into 4 clusters (if that yields an AMI with true labels) to see if our VQ does better than a generic clustering.
* Silhouette score, we’ll just report the value. Typically, >0.5 is considered a fairly strong clustering, 0.25-0.5 moderate, <0.2 weak.
* Temporal consistency: We will average the “consistency score” (like fraction of dominant state per segment) across all segments for each activity type. We can provide mean ± std for, say, walking segments had 90% ± 10% time in one state; running segments had 85% ± 15%, etc. If needed, a statistical comparison can be between activities (maybe transitions naturally have lower consistency than steady states – which is expected).
* If we had any doubt about whether the clustering alignment is significant, we could use a permutation test for AMI (randomly reassign codes to segments and recompute AMI to get a null distribution). But likely, if we see a decently high AMI (e.g., >0.5), that itself is convincing enough.
* **Interpretation:** There’s no single threshold for “AMI good enough,” but we will interpret results in terms of how clean the mapping is. We expect some overlap due to transition segments; as long as the main behaviors are captured, we’ll consider this validated.
* **Experiment E3 (Classification performance):**
* We will utilize cross-validation, so we’ll have multiple runs (folds). We can report the mean accuracy across folds and its standard deviation. If we do subject-wise splits, we might list accuracy per subject to see variation.
* To assess if accuracy meets our expectations, we might set a **target** (for example, we want >90% overall, and no class with <80% F1-score). These targets can be our success criteria.
* The confusion matrix will be presented likely as percentages. We will highlight if any off-diagonal is particularly large (e.g., if 15% of trot is misclassified as walk, that might be acceptable or not depending on needs – we’ll discuss it).
* For calibration, we’ll compute the ECE as described. If ECE > 0.1, we will likely apply a calibration method (like Temperature Scaling on the validation set) and then recompute ECE to show improvement. We’ll include both pre- and post-calibration ECE if applicable.
* We might also calculate the Brier score for completeness. A low Brier (close to 0) is good.
* If the classifier training involves any randomness (like random forest or neural net initialization), we could run it several times and show a confidence interval for accuracy. But with CV that might be overkill – CV itself gives some variance estimate.
* We will also statistically compare our classifier against a baseline (maybe a trivial baseline like always predict “Rest” or something, just to show it’s much better than chance). But given chance is 25% across 4 classes (or less if imbalanced), we expect a large improvement.
* **Experiment E4 (Uncertainty analysis):**
* The calibration curve will be plotted for qualitative assessment. To quantify, ECE is the main metric. We will likely treat ECE < 0.1 as “good calibration” per our requirement. If we have enough samples, we can compute a confidence interval for ECE via bootstrap (since ECE is a point estimate on the test set). But that’s not common; we’ll likely just quote the value.
* For correlation between entropy and errors, we’ll compute Pearson’s r. If we have, say, a correlation r > 0.5 and p < 0.01, that’s evidence of a strong positive correlation (i.e., indeed errors correlate with high entropy). If it’s lower, we interpret accordingly. Spearman rank-correlation might also be used if the relationship is not linear.
* Additionally, we will examine the cases of highest confidence errors explicitly (if any) and report what they were. This is more of an error analysis: e.g., “Out of 100 test errors, 5 had confidence > 0.9 – upon inspection, these were mostly cases where the person’s activity was mislabeled or genuinely ambiguous (like a brisk walk borderline with jog).” Such analysis helps contextualize whether confident errors are due to model flaws or data ambiguity.
* If we use reconstruction error of VQ as an uncertainty proxy, we can similarly correlate that with whether a segment was a known class or not. For example, maybe a very novel movement yields high reconstruction error and indeed doesn’t fit any known intent – that might be good (flag novel input). This is an exploratory analysis we’ll include if data permits.
* **Combining Results:** After individual analyses, we will triangulate findings across experiments. For instance, if E2 shows one particular code is “mixed” (coherence issue) and E3’s confusion matrix shows difficulties in a certain class, are they related? (Maybe that code corresponds to where the classifier also struggles – pointing to a specific behavior that’s hard to pin down). We’ll look for such alignments. Statistical evidence combined with domain reasoning will support our conclusions.
* **Reporting:** All statistical outcomes will be reported with proper context (e.g., “The 7-code model’s mean reconstruction error = 0.15g (±0.02), versus 0.16g (±0.02) for the 8-code model; p = 0.03 for difference, indicating a slight but significant improvement with 7 codes.”). We will avoid technical jargon where not needed and ensure results are linked back to the hypotheses (e.g., “thus, we have evidence to accept the hypothesis that the optimized codebook is as good as or better than the empirical one”).

This analysis plan ensures that our validation is not just anecdotal but statistically grounded. We will be cautious of over-interpreting small differences – confidence intervals and significance tests will help prevent that.

## Success Criteria and Go/No-Go Decisions

Finally, we define what outcomes of T0.5 will be considered a successful validation and what outcomes would raise red flags (potentially causing a no-go decision or a need to revamp aspects of the project). These criteria help us have an objective checkpoint before moving on to the next phase:

* **Discrete-State Representation Success:**
* **Criteria:** The FSQ/VQ codebook demonstrates meaningful compression of behavior. Specifically: reconstruction error for the chosen codebook is not significantly worse than alternatives (and ideally is better or equal with fewer codes), and the code utilization is reasonable (we don’t necessarily need 100% utilization, but we should see that each of the top-level codes corresponds to a subset of data and not have, say, half the codes completely unused). We also expect to identify at least some codes that clearly map to the intended behaviors (Rest/Walk/Trot).
* **Go if:** The codebook [7,6,5,5,4] shows equal or better performance compared to [8,6,5,5,4], cluster metrics show distinct behavior groupings (e.g., AMI well above chance, at least ~0.5+ for coarse behaviors), and temporal consistency is high (most steady activity segments dominated by one state). Low code utilization alone isn’t a showstopper if those used codes cover all needed behaviors, but we should understand why (e.g., maybe our dataset just didn’t invoke the other codes yet).
* **No-Go if:** Reconstruction error or classification drops noticeably with the chosen codebook (meaning we harmed performance), or if the discrete states appear largely incoherent (e.g., every code is a mix of multiple behaviors, AMI near zero). If the discrete representation doesn’t clearly work, then proceeding to build on it would be unwise. We’d need to reconsider the model (maybe more training, different hyperparameters, or in worst case, question the discrete assumption itself). A no-go might also be triggered if only a tiny fraction of codes (<5%) are used and we find that, say, important behavior distinctions (like walk vs trot) are *not* actually separated by codes (meaning the model failed to differentiate them). In that case, we cannot trust the model to handle different behaviors distinctly.
* **Behavioral Alignment (Intent Mapping) Success:**
* **Criteria:** The human-to-quadruped mapping via intent classification works accurately and robustly. Specifically: overall intent classification accuracy meets a predetermined threshold (for instance, **≥90%** overall, with no critical class below ~80% accuracy or F1), and the model’s confidence is well-calibrated (ECE ≤ 0.1). Moreover, the mapping of human activities to robot behaviors should cover the key scenarios without mismatch.
* **Go if:** The classifier performs well in cross-validation, confusion matrix is acceptable (e.g., minimal confusion between rest and moving states, some manageable confusion between walk/trot if any), and we have high confidence in its predictions (and where confidence is low, we know it). If ECE is slightly above 0.1 initially but can be remedied with calibration, that’s fine. The key is we end up with a system that can say “I’m X% sure the human intent is Walk” and be usually right. Additionally, if any real quadruped data was tested, the system’s decisions for those data should make sense (e.g., when we fed in dog IMU for trotting, the system indeed chose “Trot” or at least had that among top states). Even qualitative validation from an expert that “yes, this looks reasonable” on real data would be a pass here given limited samples.
* **No-Go if:** The classification accuracy is low (for example <75%, meaning it’s often wrong or too confused) – that would indicate our approach might not generalize or features are inadequate. Also, if calibration is wildly off (ECE >> 0.1) and cannot be easily fixed, that’s a trust issue. Another no-go would be if, when trying it (even in simulation or a simple real test), the robot would clearly execute wrong behaviors (e.g., our system consistently interprets a human running as the robot should “Walk” – a reversed mapping). That would show a fundamental mapping problem. If such issues arise, we’d have to either retrain with better data, refine feature engineering, or even reconsider our intent categories.
* **Real Data Transfer Success:**
* **Criteria:** Even though our system is trained on human data, it should ideally show some validity on real quadruped data if provided. This is more of a bonus criterion at T0.5 (since we might not have much real data yet), but it is important for declaring the approach viable. For example, if we recorded a dog walking and fed that into the system, it should classify it as “Walk” (perhaps with some uncertainty if domain shift, but at least not classify it as something completely off like “Rest”).
* **Go if:** In any limited test with real quadruped or high-fidelity simulated data, the system’s outputs are qualitatively correct (or at least not obviously wrong). Essentially, no red flags in transfer: the features from quadruped data seem to fall into the right regions of our model’s space.
* **No-Go if:** Testing on even one or two real quadruped examples produces nonsensical results (e.g., the dog trotting data was classified as “Rest” with high confidence). That would imply a severe domain mismatch. It doesn’t necessarily kill the project, but it means we cannot proceed without addressing that gap – likely by incorporating real quadruped data into training or doing domain adaptation.
* **Uncertainty and Safety Success:**
* **Criteria:** The system is capable of identifying when it doesn’t know. This is slightly harder to quantify as a strict go/no-go, but we want to see at least that **most failure cases come with low confidence**.
* **Go if:** In the evaluation, a high percentage of errors were flagged by high uncertainty (say, >80% of the misclassifications had below 50% confidence or high entropy). This means the system rarely makes an error while being confident – which is good for safety. Also, if we set a confidence threshold (like “only act autonomously if confidence > 0.8”), the number of decisions that fall above that threshold and are correct should be high, and the ones below we can handle separately (like ask for human confirmation). If such a threshold can be found that yields a good trade-off, that’s a success.
* **No-Go if:** The system frequently is *confidently wrong* (e.g., many errors had confidence > 0.8). That is dangerous because the system would mislead us into trusting it when it’s wrong. If calibration fixes this, fine; if not, the model might be fundamentally overfit or under-expressive. We would then need to either improve the model or incorporate an explicit uncertainty model (like Bayesian approach or ensemble) for safety before deployment.
* **Documentation and Transparency:** (Another success criterion for ourselves) We should have all the validation findings well-documented, with evidence. The protocol is only successful if it yields clear answers to the posed questions. So as a meta-criterion:
* **Go if:** We can, at the end of T0.5, hand over a validation report that says, for each hypothesis “validated” or “not validated” with supporting data.
* **No-Go if:** Results are inconclusive or too variable to draw firm conclusions (which might indicate not enough data or too high noise – meaning we might need more experiments).

In summary, a **successful T0.5 validation** will give us confidence that: 1. Our discrete behavior states are a valid abstraction (with evidence of representing real behaviors distinctly), 2. Our human-to-robot behavior mapping works accurately (with the model correctly inferring intents and being aware of its uncertainty), and 3. There are no major unknowns left regarding whether this will work on a robot (aside from implementation details to be handled in later stages).

If all criteria are met (or those not fully met have a mitigation plan), we can proceed to integrate and perhaps live-test the system in T1. If one of the core criteria fails, we will pause and address it – whether that means gathering more data, changing model parameters, or in a worst-case scenario, reconsidering the approach (for example, if the discrete hypothesis utterly failed, maybe we’d consider a continuous state model or a larger network without quantization).

**Go/No-Go Decision Meeting:** We will hold a review meeting after all experiments are done. We will go through each success criterion with the data and decide go or no-go. For any “no-go” outcome, we will outline what needs to be done (repeat experiments with fixes, gather more data, etc.) before we can move forward. Only when all critical issues are resolved will we exit the T0.5 phase.

By following this validation protocol, we aim to thoroughly vet our system’s core assumptions in a structured manner. This reduces the risk in subsequent phases and builds confidence with stakeholders (and ourselves) that the approach is scientifically sound and practically feasible. The experiments and criteria defined here will guide the team through T0.5 and ensure we have measurable evidence to support the system’s design decisions.