POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Layer-wise Transferability of Pre-generation Success Probes in Multilingual Large Language Models

Tesi di Laurea Magistrale in Computer Science and Engineering - Ingegneria Informatica

Author: **Andrea Paganelli**

Student ID: 250277
Advisor: Prof. Mark Carman
Co-advisor: Prof. Gianluca Demartini
Academic Year: 2025-26

Abstract

Large language models (LLMs) are increasingly used in multilingual settings, where reliability and inference cost vary across models, languages, and prompts. Recent work has shown that a model's future success can be predicted from pre-generation hidden activations, but it remains unclear whether these signals can be reused across languages without labelled examples in each target language.

This thesis studies the layer-wise transferability of pre-generation success probes in multilingual large language models. A multilingual benchmark is constructed from 3,000 MATH problems translated into ten languages. For each model, language, and problem, repeated generations are used to estimate empirical success rates, while pre-generation residual-stream activations are extracted at every layer. Linear ridge-regression probes are then trained to predict model success under same-language, cross-lingual, and pooled multilingual settings.

The results show that success is linearly decodable across models and languages, and that cross-lingual transfer is consistently informative. However, transfer depends on layer depth, with the most transferable signals generally appearing in intermediate or middle-to-late layers and declining toward the final layers. English-trained probes transfer discriminatively but are often poorly calibrated, while pooled multilingual probes provide more reliable score estimates. A routing simulation further shows that pooled multilingual probes achieve a stronger accuracy–cost trade-off than English-only transfer probes. Overall, the thesis shows that pre-generation success signals can transfer across languages, but reliable multilingual use requires attention to layer depth, calibration, and decision thresholds.

Keywords: Artificial Intelligence; Large Language Models; Natural Language Processing; Mechanistic Interpretability; LLM Routing.

Sommario

I Large Language Models (LLMs) sono sempre più utilizzati in contesti multilingue, dove l'affidabilità delle risposte e il costo di inferenza possono variare sensibilmente in funzione del modello utilizzato e alla lingua in cui è scritto il prompt. Studi recenti hanno dimostrato che è possibile prevedere il futuro successo di un modello analizzandone le attivazioni interne prima che inizi a generare una risposta. Tuttavia, non è chiaro se questi segnali possano essere trasferiti e riutilizzati tra lingue diverse.

Questa tesi analizza tali segnali attraverso l'uso di probe, ovvero semplici modelli predittivi addestrati sulle attivazioni interne prodotte dopo l'elaborazione del prompt. L'obiettivo è comprendere se i segnali appresi in una lingua possano essere riutilizzati anche in altre lingue, oppure se sia necessario disporre di dati etichettati specifici per ciascuna lingua di destinazione.

A questo scopo, viene costruito un benchmark multilingue basato su 3.000 problemi del dataset MATH tradotti in dieci lingue. Per ogni modello, lingua e problema vengono generate più risposte, in modo da stimare un tasso empirico di successo. Le attivazioni vengono quindi estratte a diversi livelli del modello e utilizzate per addestrare regressori lineari con l'obiettivo di predire la correttezza della risposta finale.

I risultati mostrano che è possibile prevedere il successo di un modello a partire dalle attivazioni estratte prima dell'inizio della generazione, e che i probe mantengono capacità predittiva anche quando vengono trasferiti tra lingue diverse. Tuttavia, la trasferibilità varia attraverso la profondità del modello, con segnali più riutilizzabili presenti nei layer intermedi. Inoltre, i probe addestrati esclusivamente in inglese risultano spesso informativi ma scarsamente calibrati, mentre quelli addestrati su dati multilingue producono stime più affidabili. Infine, una simulazione di routing mostra che questi segnali possono essere impiegati per selezionare in modo efficiente il modello da utilizzare per ciascun prompt.

Parole chiave: Intelligenza Artificiale; Large Language Models; Elaborazione del Linguaggio Naturale; Interpretabilità; LLM Routing.

Contents

1 Introduction

Large language models are among the most influential recent developments in artificial intelligence. They are increasingly integrated into digital tools, services and professional workflows, where they support the processing, generation and interpretation of language across a wide range of contexts.

1.1. Motivation

As the use of LLMs expands, their reliability becomes increasingly important, since these systems are expected to produce useful outputs across diverse inputs, languages, and levels of task difficulty.

This expectation is difficult to satisfy because model performance is not uniform. A language model may solve one problem correctly while failing on another that appears similar, and the same underlying task may become easier or harder when expressed in a different language. These differences matter in practical applications because unreliable outputs can reduce trust in the system, while using the strongest available model for every input can make deployment unnecessarily expensive.

This creates a trade-off between reliability and cost. Stronger models can improve expected performance, but they may be unnecessary for inputs that a smaller or cheaper model could already answer correctly. Conversely, cheaper models reduce inference cost, but may fail more often on difficult prompts or in languages where their competence is weaker. A useful system would therefore benefit from estimating whether a given model is likely to answer a specific input correctly before the cost of generation is incurred.

Pre-generation success prediction offers a way to study this problem. The central idea is to estimate a model's likelihood of success from its internal activations after it has processed the prompt but before it has generated an answer. If such information is present in the hidden representation, a lightweight probe can be trained to predict future correctness without requiring the model to complete the full generation first. This makes pre-generation success prediction relevant both as an interpretability question and as a

practical mechanism for failure detection or model routing.

In multilingual settings, however, success prediction raises an additional challenge. A probe trained in one language may capture information about the underlying task, but it may also depend on language-specific features of the activation space. These features can arise from tokenisation, script, translation choices or differences in model competence across languages. As a result, a probe that works well within one language may not necessarily produce reliable predictions when applied to another.

The motivation of this thesis is to understand whether pre-generation success signals can be reused in multilingual settings. If such signals transfer discriminatively across languages, they could reduce the need to train separate probes independently for every language. However, practical reuse also depends on whether the transferred scores are sufficiently reliable for downstream decisions such as routing.

1.2. Problem Statement

Pre-generation success prediction is useful only if the learned predictor generalises beyond the examples and conditions used to train it. In a multilingual setting, this requirement is particularly demanding. A success probe may perform well when trained and evaluated on prompts written in the same language, but this does not show whether it has learned a transferable signal of task difficulty and model competence, or whether it has instead captured language-specific regularities in the model's activation space.

This thesis addresses the cross-lingual transferability of pre-generation success probes. The central question is whether a probe trained to predict success in one language can be reused in another language without additional target-language fitting. This matters because collecting labelled rollout data for every model and language is costly: it requires repeated generations, correctness checking, and probe training for each deployment setting.

The problem also has a layer-wise dimension. Transformer models process prompts through a sequence of layers, and representations at different depths may encode different types of information. Some layers may be more sensitive to language-specific surface form, while others may encode more shared semantic or task-level structure. It is therefore unclear whether transferable success information is available uniformly across model depth, or whether it is concentrated in particular layers.

A further challenge is calibration. A probe may discriminate between easier and harder examples while still producing scores whose numerical values are unreliable across lan-

guages. This distinction is important for downstream decisions, because routing policies and threshold-based systems depend not only on the relative ranking of examples, but also on the meaning of the predicted score. This thesis therefore treats discrimination and calibration as complementary aspects of multilingual success prediction.

Finally, the thesis considers whether these success estimates can support cost-aware model selection. Routing is used to test whether multilingual success probes can guide the choice between cheaper candidate models and a stronger model while preserving expected answer quality.

1.3. Research Questions

The thesis is guided by three research questions:

RQ1. To what extent do pre-generation success probes transfer across languages?

RQ2. At which model depth are cross-lingual success signals most transferable?

RQ3. To what extent can multilingual pre-generation success probes support cost-aware model routing?

The first two questions define the core empirical focus of the thesis. They examine whether probes trained in one language can be reused in other languages, and how the transferability of the success signal changes across model layers. The third question evaluates the practical relevance of these predictions through a routing simulation, where success estimates are used to guide model selection.

1.4. Contributions

This thesis contributes an empirical study of pre-generation success prediction in multilingual large language models. It extends prior work on success probing by moving from same-language prediction to a cross-lingual setting, where the same underlying reasoning problems are expressed in multiple languages and the transferability of learned probes can be evaluated directly.

The study introduces a multilingual probing setup based on translated mathematical reasoning problems. By keeping the underlying problem fixed while varying the prompt language, the experimental design supports an analysis of whether success prediction reflects task-level difficulty, language-dependent effects, or a combination of both.

A central contribution is the layer-wise analysis of success prediction and cross-lingual

transfer. By evaluating probes across model depth, the thesis identifies where success-related information is most accessible and where it transfers most effectively across languages.

The thesis also distinguishes between discriminative transfer and calibrated score transfer. It evaluates not only whether probes can rank successful and unsuccessful examples, but also whether the predicted scores remain reliable across languages. This distinction is important because practical uses of success prediction, such as routing, require scores that have a consistent interpretation across target languages.

As an applied validation, the thesis includes a cost-aware routing case study. This experiment evaluates whether success probes can support model-selection decisions between cheaper models and a stronger model. The case study connects multilingual success prediction to an applied inference problem, while keeping the main contribution focused on the transferability and reliability of pre-generation success signals.

Finally, the thesis evaluates the same probing approach on BFCL, a structured function-calling benchmark. This cross-dataset evaluation tests whether pre-generation success prediction can also be applied outside mathematical reasoning when supervision from the target dataset is available.

1.5. Scope and Assumptions

The scope of this thesis is pre-generation success prediction in structured evaluation settings. The main experiments focus on multilingual mathematical reasoning problems, where the same underlying problem can be evaluated across translated prompts. The thesis also evaluates function calling through BFCL, but this setting is used only as an English cross-dataset evaluation and not as part of the main multilingual analysis. The conclusions should therefore be interpreted primarily with respect to multilingual mathematical reasoning, with BFCL providing limited evidence that the same probing approach can also be applied to another structured task.

The analysis focuses on hidden activations extracted after prompt processing and before answer generation. It does not study mid-generation states, token-level uncertainty during decoding, or post-generation self-evaluation. This choice reflects the operational motivation of the work: a pre-generation predictor can support decisions before the cost of generating a full answer is incurred.

The probing method is intentionally simple. The thesis uses linear ridge-regression probes trained on standardised hidden activations. This choice tests whether success information

is linearly accessible from the representation, rather than attempting to build the most powerful possible predictor. The resulting evidence should therefore be understood as evidence about linear decodability and transferability, not as an upper bound on achievable prediction performance.

The multilingual setting assumes that translated versions of each problem preserve the same underlying mathematical task. Translation quality is therefore important, but the thesis does not assume that translations are perfect or that all language versions are equally easy for either humans or models. Differences in script, tokenisation, phrasing, and model-language competence may influence the observed results.

The routing experiment is a simulation over already-generated rollouts, not a production deployment. It uses empirical success scores from generated answers to estimate the expected success of routing decisions. The cost model is also simplified: it uses fixed model prices and expected output-generation costs, and it excludes latency, batching effects and hardware utilisation. The routing results should therefore be interpreted as a controlled comparison of routing policies rather than as a complete deployment cost analysis.

Overall, these choices define the thesis as a controlled empirical study of multilingual pre-generation success prediction. The results should therefore be interpreted as evidence in the evaluated setting, rather than as a general claim about all language models, tasks or deployment environments.

2 | Background and Related Work

This chapter establishes the conceptual and methodological foundations of the thesis. It first introduces large language models, tokenisation, hidden states, and layer-wise representations, which define the internal activations analysed in the experiments. It then reviews representation probing, multilingual transfer, success prediction, calibration, and model routing. The chapter concludes by positioning the thesis within the existing literature and identifying the open question of whether success-related information in pre-generation activations can transfer across languages and support reliable cost-aware decisions.

2.1. Large Language Models and Pre-generation Representations

Large language models are neural models that process and generate text by representing language as sequences of tokens. These tokens are mapped to numerical representations and transformed through multiple layers of computation before the model produces a prediction for the next token. The internal vectors produced during this process are commonly referred to as hidden states, and they provide the basis for analysing how information is represented inside the model.

This section introduces the background concepts needed to understand these internal representations. It first describes the language modelling objective and autoregressive generation, then discusses tokenisation and multilingual input representation. It then introduces decoder-only transformers, hidden states, and layer-wise representations, which are the internal model states used in the probing experiments later in the thesis.

2.1.1. Language Modelling and Autoregressive Generation

Large language models are commonly trained with a language modelling objective. In an autoregressive setting, the model receives a sequence of tokens and learns to predict the next token from the preceding context. Given a sequence $x_1, x_2, \ldots, x_T$, the probability

of the full sequence can be decomposed as

$$P(x_1, x_2, \ldots, x_T) = \prod_{t=1}^{T} P(x_t \mid x_1, \ldots, x_{t-1}).$$

This objective provides a dense learning signal because each position in a text sequence can be used as a prediction target. During training, the model adjusts its parameters so that the probability assigned to observed continuations increases. Over large-scale text datasets, this process allows the model to learn statistical, linguistic, and semantic regularities that can later support tasks such as question answering, translation, summarisation, and mathematical reasoning (Bengio et al., 2003; Brown et al., 2020).

At inference time, the same autoregressive structure is used to generate text token by token. The prompt is first encoded as a sequence of input tokens. The model then computes hidden representations for this prompt and predicts a distribution over the next token. A decoding strategy selects a token from this distribution, appends it to the context, and repeats the process until a stopping condition is met. The final response is therefore not produced in a single step, but through a sequence of conditional predictions in which each generated token influences the following ones (Holtzman et al., 2020).

Autoregressive generation therefore separates prompt processing from response generation. Before the first output token is selected, the model has already computed internal representations of the prompt. These representations are available before the generated reasoning trace or final answer exists, making them a natural object of analysis for studying model behaviour prior to generation.

2.1.2. Tokenisation and Multilingual Input Representation

Language models cannot operate directly on raw text. Before a prompt is processed, it must be converted into a sequence of discrete units that the model can represent numerically. These units are called tokens, and they define the basic input sequence seen by the model. Depending on the tokenizer, a token may correspond to a word, a subword fragment, a character, a number, a punctuation mark, or another recurring text pattern. Modern large language models typically use subword tokenisation, which allows frequent expressions to be represented compactly while still making it possible to represent rare or unseen words by decomposing them into smaller units (Kudo & Richardson, 2018; Sennrich et al., 2016).

Once the text has been tokenised, each token is mapped to a numerical vector representation. These vectors are usually referred to as token embeddings. They provide the first

learned representation of the input sequence and allow the model to process discrete symbols using continuous numerical computation. Unlike one-hot representations, which only identify tokens as separate symbols, embeddings can capture distributional relationships between tokens. Tokens that occur in similar contexts can therefore acquire related representations, allowing the model to generalise across words, subword units, and symbols that appear in similar linguistic environments (Bengio et al., 2003; Mikolov et al., 2013).

A token representation also needs information about position. The same token can play different roles depending on where it appears in a sequence, and changing the order of tokens can change the meaning of the text. This is especially important in mathematical prompts, where the position of numbers, variables, operators, and constraints can determine the structure of the problem. Positional information therefore allows the model to distinguish between the same token appearing in different locations and to construct representations that depend on both token identity and sequence order (Dufter et al., 2022; Vaswani et al., 2017).

Tokenisation is particularly important in multilingual settings. The same content can produce very different token sequences depending on the language and script in which it is written. Languages using Latin, Cyrillic, Arabic, Thai, Telugu, or Chinese scripts may be segmented differently by the same tokenizer. As a result, translated prompts can differ in token length, token boundaries, and the frequency of the subword units used to represent them. Prior work has shown that tokenizers can introduce systematic disparities across languages, with some languages requiring substantially more tokens than others to express equivalent content (Petrov et al., 2023). The suitability of a multilingual tokenizer can therefore influence model performance, especially when the vocabulary represents some languages more efficiently than others (Rust et al., 2021).

These differences mean that translated versions of the same text are not identical from the model's perspective. Even when the intended meaning is preserved, the model receives different token sequences and may construct different internal representations from them. Tokenisation therefore provides one mechanism through which language, script, and surface form can affect multilingual language model behaviour.

2.1.3. Decoder-only Transformers and Hidden States

Most contemporary generative large language models are based on decoder-only transformer architectures. A decoder-only transformer receives a sequence of tokens, constructs contextual representations for those tokens, and uses the resulting representation at the end of the sequence to predict the next token. This design matches the autoregressive gen-

eration process described above. The model predicts the next token from the tokens that precede it, and generated tokens are appended to the context before the next prediction is made (Brown et al., 2020; Vaswani et al., 2017).

The central operation in a transformer is self-attention. Self-attention allows the representation of each token to be updated using information from other tokens in the sequence. Rather than representing each token in isolation, the model can assign different levels of importance to different previous tokens and combine information from them when constructing a contextual representation. This is important because the meaning of a word, symbol, or mathematical expression often depends on its surrounding context. In a mathematical prompt, for example, a number or variable may only become meaningful once it is interpreted together with the surrounding constraints and question statement.

In decoder-only models, self-attention is constrained by causal masking. Causal masking prevents a token from attending to future positions in the sequence. During training and generation, this ensures that the model only uses information that would be available in a left-to-right prediction setting. When the input consists only of the prompt, the final prompt token is the last position in the sequence and can attend to the entire prompt. Its representation is therefore a compact internal state formed after the model has processed the input but before any output token has been generated.

The representations produced inside the model are called hidden states. A hidden state is a vector associated with a token at a particular layer of the model. Unlike the initial token embedding, which mainly represents the token before contextual processing, a hidden state reflects information gathered from the surrounding context through the transformer layers. The same token can therefore have different hidden states depending on the prompt in which it appears. Hidden states are the internal representations that make it possible to study what information is available inside the model before observing its final output.

Each transformer layer updates these hidden states through a combination of attention, feed-forward transformations, residual connections, and layer normalisation. Attention allows information to be exchanged across token positions, while feed-forward networks transform the representation at each position. Residual connections help preserve and update information across layers, and layer normalisation stabilises the scale of the activations during computation (Ba et al., 2016; Vaswani et al., 2017). Together, these operations produce a sequence of progressively transformed hidden states as the prompt passes through the model.

2.1.4. Layer-wise Representations and the Residual Stream

A transformer does not produce a single representation of the input. Instead, the prompt is transformed repeatedly as it passes through the model's layers. Each layer receives the hidden states produced by the previous layer and updates them through attention and feed-forward computation. The result is a sequence of layer-wise representations, where each layer provides a different internal description of the same input sequence.

These layer-wise representations are important because different kinds of information may become more or less accessible at different depths. Earlier layers are often closer to the surface form of the input, including local token patterns and lexical information. Later layers may encode more contextual and task-relevant structure, since they are produced after multiple rounds of information mixing and transformation. Prior work on transformer representations has shown that linguistic and semantic properties can vary across layers, suggesting that internal information is distributed across model depth rather than being concentrated only in the final layer (Jawahar et al., 2019; Tenney et al., 2019).

The residual stream provides a useful way to understand how information is carried through a transformer. Each layer updates an existing representation rather than replacing it entirely. Residual connections allow information from earlier layers to persist while new information is added by attention and feed-forward operations. The hidden state at a given layer can therefore be viewed as one point in an evolving internal representation of the prompt (J. Song et al., 2025; Vaswani et al., 2017).

Layer-wise analysis is especially useful when studying internal model behaviour. If a property is encoded in the model's hidden states, it may not be equally accessible at every layer. A probe trained on an early layer may capture different information from a probe trained on a later layer. Examining representations across depth therefore makes it possible to study not only whether information is present, but also where it becomes most accessible inside the model.

2.2. Representation Probing in Language Models

Hidden states provide an internal representation of the input, but they are not directly interpretable on their own. Representation probing is a method for studying these hidden states by testing whether a particular property can be predicted from them. Instead of analysing only the model's generated output, probing examines what information is accessible from intermediate activations inside the model.

This section introduces probing as a general method for analysing language model rep-

resentations. It first explains how probes are used to test for recoverable information in hidden states. It then discusses linear and regression-based probes, which are the type of probe used in this thesis. Finally, it outlines the main limitations of probe-based evidence, especially the distinction between information being decodable from a representation and that information being causally used by the model.

2.2.1. Probing as a Method for Studying Representations

A probe is a supervised model trained to predict a property from hidden representations. In neural language models, the input to the probe is usually an activation vector extracted from a particular layer, while the prediction target is a property of interest. If the probe can predict that property from the representation, this suggests that the information is present, or at least recoverable, from the model's internal state. Probing is therefore a way of studying what information hidden representations make available, rather than only evaluating the final output of the model (Alain & Bengio, 2016; Belinkov, 2022).

Probing has often been used to study linguistic properties encoded in language models. For example, probes can be trained to predict syntactic structure, semantic roles, or other properties associated with natural language processing tasks. Work on contextual language models has shown that different linguistic properties can often be recovered from different layers, suggesting that model representations contain structured information beyond the surface token sequence (Belinkov, 2022; Tenney et al., 2019). More broadly, probing does not need to be limited to linguistic properties of the input. It can also be used to study behavioural properties of the model, including information associated with later decisions or outputs.

The interpretation of probing results requires care. Strong probe performance suggests that the target property is encoded in the representation in a way that the probe can recover. This can reveal information that is not directly visible from the text alone, because the probe is trained on internal model states rather than on the raw input sequence. At the same time, probing should be understood as representational evidence rather than a complete explanation of model behaviour. A successful probe shows that information is decodable from a hidden state, but it does not by itself establish how the model uses that information during generation (Belinkov, 2022; Hewitt & Liang, 2019).

2.2.2. Linear Probes

A linear probe is a probe that predicts a target using a weighted combination of hidden-state features. Given a hidden representation, the probe applies a linear mapping from

that representation to the target label or score. This restriction is important because it tests whether the target property is linearly accessible from the representation. Compared with a more complex nonlinear classifier, a linear probe provides a stricter and more interpretable test: good performance suggests that the relevant information is available in a relatively simple form within the hidden state (Alain & Bengio, 2016).

When a linear probe performs well, the result is often interpreted as evidence that the target property is represented along a direction or subspace of the activation space. This does not mean that the model explicitly identifies or uses that direction in the same way as the probe. Rather, it means that the information can be decoded from the representation by a simple external model. This distinction is important because probing concerns extractable information, whereas claims about how the model actually uses that information require stronger forms of evidence (Belinkov, 2022).

The simplicity of linear probes is one reason they are common in representational analysis. If the probe is too expressive, it may learn the prediction task itself instead of revealing information already encoded in the model representation. Simpler probes reduce this risk by limiting the amount of computation performed outside the original model. This makes the analysis more focused on the structure of the hidden representation and less dependent on the capacity of the probe. For this reason, probe complexity is an important consideration when interpreting probing results (Alain & Bengio, 2016; Hewitt & Liang, 2019).

2.2.3. Regression-based Probes

Probes can be formulated either as classification models or as regression models, depending on the type of property being predicted. Classification probes predict discrete labels, such as whether a sentence has a particular syntactic structure or whether an example belongs to one of several categories. This formulation is appropriate when the target property is naturally binary or categorical. Regression probes, by contrast, predict continuous values. They are useful when the property of interest is not simply present or absent, but varies along a scale. This distinction matters because probing is not limited to recovering linguistic labels from representations; it can also be used to study graded properties of model behaviour (Belinkov, 2022).

Many behaviours of language models are better described as degrees rather than fixed binary outcomes. A model does not necessarily have a deterministic success or failure state for a given prompt, especially when generation involves sampling. Instead, success can be understood probabilistically: the same model may answer correctly in some generations

and incorrectly in others. In such settings, a binary probe can only separate examples into successful and unsuccessful cases, while a regression probe can represent a more gradual notion of expected success. This makes regression-based probing useful when the target reflects the strength, probability, or reliability of a behavioural property rather than a categorical label. Related work on probing has similarly argued that evaluation should consider not only whether information can be extracted, but also how much information is accessible and how difficult it is to recover from a representation (Voita & Titov, 2020).

Regularised linear regression provides a simple way to predict continuous targets while controlling probe complexity. Regularisation adds a penalty to the probe objective so that overly large or unstable weights are discouraged. This is important because hidden representations in modern language models are high-dimensional, and an unconstrained probe may overfit to incidental correlations in the training data. By limiting the capacity of the probe, regularisation helps preserve the intended role of probing: testing whether the target information is accessible from the representation, rather than allowing the probe to solve the task independently. This concern is central in the interpretation of probing experiments, where overly expressive probes can achieve strong performance even when the representation itself is not meaningfully organised around the target property (Hewitt & Liang, 2019).

2.2.4. Limits of Probe-based Evidence

The main limitation of probing is that decodability is not the same as causality. If a probe successfully predicts a property from a hidden representation, this shows that the property is recoverable from that representation by the external probe. It does not show that the original model uses that information when producing its output. For example, a hidden state may contain information correlated with a later decision, while the model's actual computation relies on different features. Establishing that a representation is causally involved in model behaviour requires stronger evidence, such as interventions that remove, modify, or control the relevant information and then measure the effect on the model's outputs (Elazar et al., 2021).

Probe performance can also be affected by probe complexity and dataset artefacts. High-dimensional hidden states may contain many incidental signals, including correlations that are not central to the model's behaviour. If the probe is sufficiently flexible, it may exploit these signals or memorise patterns in the probing dataset. This can make the representation appear more informative than it actually is for explaining the model's computation. Control tasks and restrictions on probe capacity have therefore been proposed as ways

to test whether probe performance reflects meaningful structure in the representation or merely the ability of the probe to fit superficial regularities (Hewitt & Liang, 2019). More recent work also cautions that high probing accuracy does not necessarily explain downstream model performance, since probes may identify signals that are predictive in the dataset but not mechanistically responsible for the model's behaviour (A. Agarwal et al., 2025).

For these reasons, probe results should be interpreted as evidence about accessible information in representations, not as complete explanations of model behaviour. A successful probe can show that a representation contains information related to syntax, semantics, difficulty, or future success, but it does not by itself explain how that information is used internally. This distinction is especially important when probing behavioural properties such as model success or failure. In such cases, the probe may reveal that the model state contains a signal correlated with future correctness, while leaving open the separate question of whether this signal is used by the model during generation. Careful interpretation therefore treats probing as a diagnostic tool for analysing representations, rather than as a direct causal account of the model's decision process (Belinkov, 2022; Voita & Titov, 2020).

2.3. Multilingual Models and Cross-lingual Transfer

A translated prompt can preserve the same underlying task while changing almost everything the model directly sees: the script, the token sequence, the prompt length, and the linguistic form. Multilingual transfer depends on whether the model nevertheless maps these different inputs to representations that share useful structure. This section discusses the tension between shared semantic representations and language-specific shifts, leading to the question of whether a probe trained in one language can remain valid in another.

2.3.1. Multilingual Language Models

Multilingual large language models are trained on text from multiple languages, allowing them to process and generate text across different linguistic systems. Unlike monolingual models, which are primarily optimised for one language, multilingual models are exposed to a mixture of languages during training and can often respond to prompts written in different scripts, vocabularies, and grammatical structures. This makes them useful in settings where users interact with the same model across languages. However, multilingual training does not imply uniform multilingual competence: a model may support many languages at the interface level while still performing substantially better in some

languages than in others (Qin et al., 2025).

One reason for this variation is that languages are not equally represented in large-scale training data. High-resource languages, especially English and other widely available web languages, typically appear more frequently and with greater domain diversity than lower-resource languages. As a result, the model may receive stronger supervision for some languages during pretraining, while other languages are represented by smaller, noisier, or less diverse text collections. This imbalance can lead to uneven performance across languages and has been discussed in terms of the difficulty of distributing a fixed model capacity across many languages (Conneau et al., 2020; Lupaşcu et al., 2026). In multilingual models, therefore, language coverage is not only a question of whether a language is included, but also of how well it is represented and learned.

The behaviour of a multilingual model can consequently vary even when the underlying task is similar across languages. A mathematical problem, factual question, or reasoning prompt may be translated into several languages, but the model may not process each version in an equivalent way. Differences can arise from the amount and quality of training data, from how the tokenizer segments the text, from script-specific patterns, and from language-specific associations learned during training. These factors make multilingual evaluation more complex than simply translating a benchmark and assuming that each translated example is interchangeable with the original. Instead, each language version may interact differently with the model's internal representations and generation behaviour (Pires et al., 2019).

2.3.2. Shared Semantic Representations

Despite these sources of variation, multilingual models may also learn representations that are partly shared across languages. If two inputs express similar meanings in different languages, the model may map them to related internal representations, even though their token sequences are different. This possibility is central to cross-lingual generalisation: if translated versions of the same content occupy related regions of representation space, then information learned in one language may remain useful in another. Prior work on multilingual representation learning has shown that models can exhibit cross-lingual alignment, suggesting that jointly trained multilingual systems can develop shared representational structure across languages (Conneau et al., 2020; Pires et al., 2019).

The notion of shared representations relies on the model learning features that are able to capture meaning beyond the exact words or symbols used in the input. A multilingual model does not need to treat translated sentences as identical token sequences in order

to generalise across them. Instead, it may encode aspects of their meaning, structure, or task requirements in a way that is partially independent of the particular language used to express them. This is especially relevant for translated tasks, where different surface forms correspond to the same underlying problem. In such cases, cross-lingual transfer is possible if the model's internal states preserve information about the shared task content, rather than only reflecting language-specific wording or script-level patterns (Conneau et al., 2020).

Shared representations provide one explanation for why a predictor learned in one language might remain useful in another. If the relevant information is aligned at the representation level, then a probe trained on hidden states from one language may still identify similar patterns in hidden states from another language. This does not require the prompts to have the same tokens, the same length, or the same syntactic form. It requires that the property being predicted is encoded in a sufficiently similar way across languages. In the context of cross-lingual probing, the key question is therefore not whether the input strings are identical, but whether the internal representations preserve the same predictive structure across languages (Conneau et al., 2020; Pires et al., 2019).

2.3.3. Language-specific Representation Shifts

Shared representations do not remove all language-specific variation from multilingual models. Languages differ in their writing systems, including Latin, Cyrillic, Arabic, Thai, Telugu, and Chinese scripts. These differences affect the surface form of the input before the model has processed its meaning. They may also influence how the tokenizer segments the text and how the model represents the resulting sequence. Script therefore provides one source of language-specific variation: two translated prompts may express the same content while still entering the model as very different sequences of symbols (Pires et al., 2019).

Tokenisation is another important source of variation. The same semantic content can be split into different token sequences depending on the language, script, and vocabulary of the model. Some languages may be represented with compact tokens, while others may be broken into smaller subword units or longer token sequences. These differences can affect prompt length, token boundaries, and subword composition. Since transformer hidden states are constructed from token-level representations, tokenisation differences may influence the internal states on which later predictions or probes depend (Rust et al., 2021).

Translation can also introduce variation even when the intended meaning is preserved.

A translated input may use different wording, syntactic structure, or emphasis from the original sentence. In many tasks, these differences may be minor, but in reasoning problems they can affect how difficult the prompt appears to the model. A mathematical question, for example, may preserve the same formal solution while changing the surrounding natural-language explanation. Translation is therefore not only a preprocessing step, but also a possible source of representational shift between language versions of the same task (Unanue et al., 2023).

Finally, cross-lingual differences may reflect the model's competence in each language. A task may appear more difficult in one language not because the underlying reasoning problem is harder, but because the model has weaker language-specific ability in that language. This competence can depend on the amount of training data, the quality of that data, the model's capacity, and the way different languages share or compete for representational resources. As a result, multilingual evaluation must consider the interaction between task difficulty and model-language competence, rather than treating performance differences as properties of the task alone (Blevins et al., 2022, 2024).

2.3.4. Shared and Language-specific Representations

Cross-lingual transfer depends on the balance between shared meaning and language-specific form. On one hand, multilingual models may preserve aspects of meaning, task structure, and semantic content across translated inputs. On the other hand, the same representations may also retain information about the language in which the input is written. This creates a central tension in multilingual representation learning: transfer is possible when the relevant information is shared across languages, but it can degrade when the representation is strongly shaped by language-specific surface patterns (De Varda & Marelli, 2023; Libovický et al., 2020). Recent work on multilingual difficulty representations makes this tension especially relevant for probing. Civelli et al. (2026) study linear probes for model-internal difficulty representations across translated reasoning tasks and show that the balance between same-language prediction and cross-lingual generalisation depends strongly on layer depth. Their results suggest that difficulty-related information is not uniformly shared across languages throughout the network, but changes in how transferable it is as representations are transformed across layers.

This is relevant whenever a model or an external predictor is expected to generalise across languages. If the information needed for a task is represented in a language-independent way, then a method learned in one language may remain useful in another. If the information is instead tied to language-specific activation patterns, transfer may

degrade when the input language changes. The key question is therefore whether the relevant internal structure remains stable enough across languages to support transfer.

2.3.5. Cross-lingual Probe Transfer

In the context of probing, cross-lingual transfer can be defined as training a probe on hidden representations from one source language and evaluating the trained probe on hidden representations from a different target language. The probe is not retrained on the target language; instead, its performance shows whether the source-language pattern generalises to the target language. If performance remains strong, this suggests that the relevant information is encoded in a similar way across the two languages. If performance drops substantially, this suggests that the learned predictive direction may be language-specific (Conneau et al., 2020; Pires et al., 2019).

This notion of probe transfer is related to, but distinct from, model transfer. Model transfer asks whether the language model itself can perform a task in another language. Probe transfer asks whether an external predictor trained on internal states from one language can predict a property from internal states in another. A multilingual model may perform a task across languages while still representing the relevant information differently across those languages. Conversely, a probe may transfer well if the internal structure associated with the predicted property is shared, even when the surface forms of the prompts differ (Blevins et al., 2022; Libovickỳ et al., 2020).

This distinction matters because probing is often used to study whether information is represented in a reusable form. If a probe trained in one language transfers to another, this suggests that the target property is not encoded only through language-specific patterns. If it does not transfer, then the representation may require language-specific supervision or calibration. This provides the bridge to the next section, which builds on this idea by considering whether prompt-conditioned hidden representations contain information about one specific future behaviour: whether the model is likely to answer correctly before generation begins.

2.4. Pre-generation Success Prediction

The probes studied in this thesis are trained to predict a behavioural property of the model: whether it is likely to answer correctly. Crucially, this prediction is made before generation begins, using the hidden state formed after the model has read the prompt but before it has produced a response. This section defines success prediction, explains

why the pre-generation setting is useful, and relates it to prior work on difficulty, failure prediction, confidence estimation, and verification.

2.4.1. Success Prediction as a Task

Success prediction is the task of estimating whether a model will answer a given prompt correctly. Unlike standard supervised prediction tasks, where the target is usually a property of the input, success prediction concerns the behaviour of a particular model on that input. The same mathematical problem may be associated with different outcomes depending on the model that attempts it. A smaller model may fail to solve a problem that a larger model answers correctly, while two models of similar size may differ because of their training data, architecture, or instruction tuning. This perspective is closely related to Lugoloobi et al. (2026), who study whether a model's internal activations contain enough information to predict whether it will answer correctly.

This makes success a model-specific quantity rather than an absolute property of the task. Difficulty is often discussed as though it belongs only to the problem, but in practice it depends on the interaction between the problem and the system solving it. A question that is easy for a strong model may be difficult for a weaker one; conversely, a problem may appear difficult to humans but be handled reliably by a model if similar patterns are well represented in its training data. Success prediction must therefore account for both the structure of the task and the competence of the model. This view is consistent with approaches that model question difficulty in relation to solver ability, rather than as a fixed item property independent of the respondent (Scarlatos et al., 2025).

Success is also naturally probabilistic. Modern language models do not always produce the same output for the same prompt, especially when decoding uses sampling. A prompt may therefore not be simply solved or unsolved in all cases. Instead, the same model may answer correctly in some decoding runs and incorrectly in others. Evaluation frameworks for generative models often account for this variability by considering repeated samples or pass-rate-style measures, particularly when outputs may vary across runs (Yeo et al., 2024). Under this view, a success predictor does not only classify prompts as easy or hard; it estimates how likely the model is to produce a correct answer under a specified generation procedure.

2.4.2. Pre-generation Prediction

Pre-generation success prediction restricts the prediction to information available before any answer token is produced. The model first processes the prompt and computes

internal hidden states. Only after this prompt-conditioning stage does it begin generating the response token by token. Kasnavieh et al. (2026) describes this stage as prefilling-time evaluation, where the model has encoded the input but has not yet produced the response. A pre-generation predictor therefore uses the internal state available at the boundary between prompt processing and answer generation. It differs from methods that inspect the generated response, the final answer, or the model's reasoning trace after the model has already begun producing output.

The pre-generation point matters because it is available before the cost of full generation is incurred. If a system can estimate that a model is unlikely to answer correctly before generation starts, it can decide to change how the prompt is handled. It might route the prompt to a stronger model, request additional computation, use a different decoding strategy, or abstain from using a cheaper model. This makes pre-generation prediction relevant for efficiency-sensitive settings, where the goal is not only to obtain correct answers but also to control the computational cost of producing them. Chen et al. (2023) motivate this broader view of cost-aware inference by showing that different models can be combined to reduce cost while preserving performance.

The possibility of pre-generation prediction depends on whether the prompt-conditioned internal state contains information about future behaviour. If hidden representations encode signals related to the model's likely success or failure, then an external predictor may be able to decode those signals before the answer is generated. This does not require the model to have already produced a solution, nor does it require the predictor to inspect generated text. It requires only that the representation formed after reading the prompt contains information correlated with the model's later correctness. Work on probing arithmetic errors provides related evidence that internal states can expose information about future mistakes before they are visible in the final answer (Sun et al., 2025).

2.4.3. Prior Work on Difficulty and Failure Prediction

Prior work suggests that language model representations can contain information related to problem difficulty. This line of work is important because difficulty is not always directly visible from the surface form of a prompt. Two problems may have similar length or wording while requiring very different reasoning steps, and two translated versions of the same problem may differ in how difficult they appear to a model. Lugoloobi and Russell (2025) show that human-labelled difficulty can be strongly decoded from LLM hidden representations using linear probes across layers and token positions. Their results support the idea that difficulty-related information may already be present internally before the

model produces an answer. If internal representations encode such information, then hidden states provide a natural place to look for signals of future success or failure.

More directly, Lugoloobi et al. (2026) show that future model success can be predicted from pre-generation activations. Their results suggest that hidden states may contain information correlated with whether the model will later answer correctly, even before the response begins. This provides the basis for success probing: rather than using hidden states only to analyse linguistic properties or input features, they can be used to predict behavioural outcomes. In this setting, the probe is trained to map prompt-conditioned activations to a measure of later correctness.

An important implication of this work is that difficulty should be understood from the model's perspective. Lugoloobi and Russell (2025) distinguish between human-labelled difficulty and LLM-derived difficulty estimates, showing that these signals do not necessarily behave in the same way. A model's internal representation of difficulty may therefore differ from human-perceived difficulty or from benchmark-level difficulty annotations. The same problem can be represented as easier or harder depending on the model's scale, training data, multilingual competence, and familiarity with the task format. This supports treating success prediction as model-specific rather than purely task-specific.

2.4.4. Different Approaches to Success Estimation

One alternative to pre-generation prediction is to ask a model to report its confidence after producing an answer. This approach is attractive because it can be applied through prompting without requiring access to internal activations. However, self-reported confidence can be poorly calibrated and sensitive to the wording of the prompt. Lin et al. (2022) show that teaching models to express uncertainty in words is possible but non-trivial, highlighting that verbalised uncertainty should not be treated as automatically reliable. A model may sound confident while being wrong, or express uncertainty despite producing a correct answer. In addition, self-reported confidence is available only after the model has generated a response, which limits its usefulness for deciding whether generation should occur in the first place.

Another family of approaches uses token probabilities or uncertainty-related signals from the generation process. These signals can provide information about how likely the model considers particular output tokens. However, high token likelihood does not necessarily imply answer correctness. A model may assign high probability to a fluent but incorrect answer, especially in tasks where correctness depends on multi-step reasoning rather than local fluency. Azaria and Mitchell (2023) further show that internal states can contain

information related to truthfulness, suggesting that correctness-related signals may not be fully captured by surface-level output probabilities.

Post-generation verification provides a different way to assess correctness. A verifier or answer checker evaluates one or more generated responses and attempts to determine whether they are correct. Such methods can improve reliability, especially in reasoning tasks where candidate solutions can be checked or compared. However, they require candidate outputs to be generated first and may add additional computational cost. Verification is therefore useful for improving or assessing completed answers, but it does not directly solve the problem of deciding in advance whether a model is worth using for a given prompt. This distinction is especially relevant in mathematical reasoning benchmarks, where answer checking is important but occurs after a candidate solution has already been produced (Fang et al., 2025).

Pre-generation prediction occupies a different point in this design space. It aims to estimate likely success before answer generation begins, using the model state obtained after processing the prompt. This makes it especially relevant for routing and cost-aware inference: if success can be estimated early, a system can decide whether to use a cheaper model, escalate to a stronger model, or apply a more expensive inference strategy before paying the cost of generation. This connects naturally to cost-aware model selection approaches such as FrugalGPT (Chen et al., 2023), but shifts the decision signal toward prompt-conditioned internal representations. The next sections build on this distinction by considering two requirements for such predictions to be useful in deployment: their scores must be reliable, and they must support decision-making under cost constraints.

2.5. Calibration and Reliability

A success score can be useful in two different ways. It may rank prompts correctly, placing likely successes above likely failures, or it may provide a probability-like estimate of success. These are not equivalent. A probe can have strong ranking performance while still producing scores that are too optimistic or too conservative. This section introduces the distinction between discrimination and calibration, which becomes essential when probe scores are used for cross-lingual transfer and routing.

2.5.1. Discrimination

Discrimination refers to the ability of a scoring model to separate positive examples from negative examples. A discriminative predictor assigns higher scores to examples that

belong to the positive class and lower scores to examples that belong to the negative class. The emphasis is therefore on relative ordering rather than on the exact numerical value of each score. This ranking-based view is closely related to receiver operating characteristic analysis, which evaluates how well a scoring function separates positive and negative cases (Fawcett, 2006).

A common measure of discrimination is the area under the receiver operating characteristic curve, or AUROC. AUROC summarises how well positive and negative examples are separated across possible decision thresholds (Bradley, 1997). This makes it useful when the final operating threshold is not known in advance, because it evaluates the quality of the ranking rather than the performance of one specific threshold. A high AUROC therefore indicates that the scoring function tends to assign higher scores to positive examples than to negative examples.

In the context of success prediction, the positive examples are prompts that the model answers correctly, while the negative examples are prompts that it answers incorrectly. A discriminative success predictor should therefore assign higher scores to prompts the model is likely to solve than to prompts the model is likely to fail. This makes discrimination useful for analysing whether hidden representations contain information that separates likely successes from likely failures. If a probe has strong discriminative performance, this suggests that the representation contains information related to the model's future behaviour.

However, a good ranking does not guarantee that the predicted scores are numerically meaningful. A predictor may assign higher scores to easier examples than to harder examples while still systematically overestimating or underestimating the true probability of success. For example, it may preserve the correct ordering of prompts but assign scores that are too high or too low in absolute terms. In that case, the predictor is useful as a ranking function but unreliable as a probability estimator. This distinction is important because downstream decisions often require scores that can be interpreted as meaningful likelihoods, not only as rankings (Guo et al., 2017).

2.5.2. Calibration

Calibration concerns the relationship between predicted probabilities and observed empirical frequencies. A calibrated predictor should assign scores that correspond to the actual likelihood of the predicted event. For example, among examples assigned a predicted probability of 0.7, the event should occur approximately 70 percent of the time. Calibration therefore asks whether the numerical value of a score has a reliable probabilistic

meaning. This is distinct from discrimination: a model can rank examples well while still producing scores that are too confident or not confident enough (Guo et al., 2017).

Expected Calibration Error, or ECE, is commonly used to summarise the mismatch between predicted confidence and observed accuracy. At a high level, it compares predicted scores with empirical outcomes and measures how far the predictions are from being reliable probabilities. A lower ECE indicates that predicted scores are closer to observed outcomes, while a higher ECE indicates that the scores are less reliable as probability estimates (Guo et al., 2017).

Reliability diagrams provide a visual way to inspect calibration. They compare predicted confidence with observed frequency, so that a perfectly calibrated predictor would lie close to the diagonal. Deviations from this diagonal indicate that the predictor is systematically overconfident or underconfident in some range of scores. Unlike a single summary measure, a reliability diagram can show where miscalibration occurs and whether it is concentrated among low-confidence or high-confidence predictions (Guo et al., 2017).

2.5.3. Calibration in Cross-lingual Settings

Calibration becomes especially important when predictions are transferred across languages. A predictor trained in one language may still rank examples usefully in another language, but its numerical score scale may not transfer cleanly. In other words, cross-lingual transfer may preserve discrimination while weakening calibration. This means that a probe trained on one language could still identify which prompts are relatively easier or harder in another language, while assigning scores that no longer correspond to the target language's actual success frequencies (Jiang et al., 2022).

Language-specific calibration shifts can arise from several sources. Different languages may produce different tokenisation patterns, prompt lengths or internal representations, even when the underlying task is semantically equivalent. The model's competence may also vary across languages depending on its training data and multilingual coverage. As a result, a score that is reliable in one language may become overconfident or underconfident in another. Prior work on zero-shot cross-lingual prediction shows that calibration can degrade under transfer, with different target languages exhibiting different calibration behaviour (Jiang et al., 2022).

This issue is central for threshold-based decision systems. Such systems do not rely only on whether one example receives a higher score than another; they also depend on the absolute value of the score. If calibration differs by language, then applying the same threshold across languages can lead to inconsistent behaviour. A threshold that

is conservative in one language may be too optimistic in another. Calibration therefore provides the bridge between success prediction and the routing setting introduced in the next section, where predicted scores are used to decide whether a prompt should be handled by a cheaper or stronger model.

2.6. Model Routing and Cost-aware Inference

Success prediction becomes operational when it is used to decide which model should answer a prompt. If a cheaper model is likely to succeed, using a larger model wastes computation; if the cheaper model is likely to fail, escalation may be worth the cost. Model routing formalises this trade-off. This section explains routing as a cost-aware inference problem and shows why calibrated pre-generation success estimates are a natural signal for multilingual routing.

2.6.1. The Model Routing Problem

Model routing refers to the problem of choosing which model should answer a given prompt. In many LLM systems, the available models differ in both capability and cost. A router may therefore choose between a cheaper model, which is faster or less expensive to run, and a stronger model, which is expected to produce higher-quality answers but requires more computation. This setting is related to the idea of LLM cascades. In both routing and cascading, the system tries to avoid using an expensive model when a cheaper model is sufficient. The difference is when the decision is made. In model routing, the system chooses one model before generation begins. In a cascade, the system may first ask a cheaper model to answer and then decide whether to escalate to a stronger model if the initial answer seems unreliable (Chen et al., 2023; Dekoninck et al., 2025).

The central difficulty in routing is the trade-off between accuracy and cost. Larger models often achieve better performance, especially on difficult prompts, but they also require more memory, computation, and inference time. Smaller models are cheaper to run, but they may fail more often on complex or unfamiliar inputs. A useful router should therefore avoid using a large model when a smaller model would have been sufficient, while still escalating prompts that the smaller model is unlikely to answer correctly. Prior work on LLM routing frames this as an optimisation problem over response quality and inference cost, where the goal is not simply to maximise accuracy, but to achieve a favourable cost-performance trade-off (Dekoninck et al., 2025; Ong et al., 2024).

Routing can therefore be understood as a prediction problem. Before choosing a model,

the system needs some signal indicating whether the cheaper model is likely to succeed on the current prompt. This signal may come from preference data, prompt features, model confidence, learned routers, or other predictive mechanisms (Ong et al., 2024). The quality of the routing decision depends on the reliability of this signal: if the router overestimates the cheap model's ability, it may route difficult prompts to a model that fails; if it underestimates the cheap model, it may unnecessarily use the expensive model. Universal routing frameworks consequently evaluate not only whether routing improves efficiency, but also whether the routing signal remains reliable across models, tasks, and benchmarks (Jitkrittum et al., 2025).

2.6.2. Routing with Success Predictors

Success predictors provide a direct signal for routing because they estimate whether a model is likely to answer correctly. In a two-model setting, a predictor can be trained to estimate the probability that the cheaper model will succeed on a given prompt. If this predicted probability is high, the prompt can be sent to the cheaper model. If it is low, the prompt can be escalated to a stronger model. This connects routing to the broader problem of matching prompts to model capabilities: different models may be appropriate for different inputs depending on prompt difficulty, task type, and the model's expected competence (W. Song et al., 2025).

A common way to convert success predictions into routing decisions is to apply a threshold. If the predicted success score is above the threshold, the router selects the cheaper model; if it is below the threshold, the router selects the stronger model. The threshold controls the trade-off between cost savings and accuracy. A lower threshold sends more prompts to the cheaper model, increasing savings but also increasing the risk of errors. A higher threshold is more conservative, preserving accuracy but reducing the opportunity for cost reduction. Dynamic thresholding and deferral rules are therefore central to routing systems, because they determine when a prompt should be handled by a cheaper model and when it should be passed to a stronger one (Ong et al., 2024; Wu et al., 2025).

This is where calibration becomes especially important. A threshold is meaningful only if the predicted score has a reliable interpretation. If a score of 0.8 does not correspond to an approximately high probability of success, then a threshold-based router may behave unpredictably. A poorly calibrated predictor may still rank prompts well, but its absolute scores may lead to overly aggressive or overly conservative routing decisions. For this reason, the distinction introduced in the previous section is crucial: discrimination may be sufficient for ranking prompts by difficulty, but calibrated scores are needed when

routing decisions depend on fixed thresholds.

Pre-generation success prediction is particularly useful for routing because it can be performed before any answer is generated. This allows the system to choose the model before spending inference compute on a full response. In contrast, post-generation confidence estimates, verifiers, or sequential cascades may require one model to generate an answer before a decision can be made about escalation. Pre-generation routing therefore offers a more direct mechanism for cost-aware inference: the system estimates whether the cheaper model is likely to succeed and immediately decides whether to use it or defer to a stronger model (Dekoninck et al., 2025; Ong et al., 2024).

This distinction matters because not all pre-generation routers obtain their routing signal in the same way. Prompt-based or embedding-based routers may make a decision from the input text alone, whereas activation-based routers require a forward pass through the model whose internal states are used as features. If activations are extracted separately from several candidate models, the router may therefore incur multiple prefill passes before selecting a single model for generation. Varshney et al. (2026) address this limitation through Encoder-Target Decoupling, where an encoder model processes the prompt and its prefill activations are used to estimate the performance of one or more target models. This separates the model used to extract routing features from the models considered for generation, extending prefill-based routing beyond the case where each target model must provide its own activations.

2.6.3. Multilingual Routing

The routing problem becomes more complex in multilingual settings. A deployed LLM system may receive prompts in many languages, and the relationship between prompt difficulty, model competence, and expected success may differ across them. A router trained on one language may therefore not behave reliably when applied to another. Even if the underlying task is similar, changes in script, tokenisation, translation quality, and model-language competence can shift the internal representations on which the router depends. Recent work on multilingual routing highlights that routing behaviour can vary across languages, making cross-lingual reliability an important concern for multilingual LLM systems (Bandarkar et al., 2025).

A further challenge is that training a router usually requires labelled examples of model success and failure. These labels are often obtained through rollouts: a model is run on a set of prompts, its answers are evaluated, and the resulting successes and failures are used to train or calibrate the router. In a multilingual deployment setting, collecting

such labelled rollouts for every target language can be expensive. The cost increases with the number of languages, models, prompts, and repeated generations needed to estimate success reliably. This creates a practical limitation for language-specific routing, especially when the system must support languages for which evaluation data is limited or costly to produce (Bandarkar et al., 2025).

A useful multilingual router should reduce the need for complete language-specific supervision. Cross-lingual and multilingual success predictors provide two general mechanisms for doing so. In a cross-lingual setting, a predictor trained using examples from one language is applied to prompts in another language. In a multilingual setting, examples from several languages are used during training so that the predictor is exposed to a broader range of linguistic variation. Both settings are relevant for routing because they aim to make the decision signal less dependent on labelled rollout data from every individual target language.

For routing, however, the usefulness of a success predictor depends on how its scores are used. If the router only compares prompts relative to one another, then a ranking signal may be sufficient. If the router applies a fixed threshold to decide whether to use a cheaper or stronger model, then the score must also be calibrated. Multilingual routing therefore brings together the two issues introduced in the previous sections: cross-lingual transfer determines whether a success signal can be reused across languages, while calibration determines whether that signal can support consistent threshold-based decisions.

2.7. Research Gap and Thesis Positioning

This section summarises the limitations of existing work and positions the thesis within the resulting research gap.

2.7.1. Limitations of Existing Work

Prior work has shown that model success and problem difficulty can be predicted from internal activations before generation. This is an important result because it suggests that the model's prompt-conditioned hidden state contains information about whether the model is likely to answer correctly, even before an output token has been produced (Lugoloobi & Russell, 2025; Lugoloobi et al., 2026). However, this evidence has largely been developed in single-language or single-distribution settings. In such settings, the probe is trained and evaluated on representations drawn from the same type of input. This leaves open a central question for multilingual use: whether the predictive signal

learned in one language remains meaningful when the same underlying task is expressed in another language. A success probe that works well within English, for example, may not necessarily work when applied to Chinese, Swahili, Arabic, Thai, or Telugu prompts.

A second limitation is that evidence of shared multilingual representations does not automatically establish cross-lingual probe transfer. Multilingual models may map semantically similar inputs in different languages to partially aligned regions of representation space, which provides a possible basis for transfer. However, this alignment can coexist with language-specific variation introduced by script, tokenisation, prompt length, translation choices, and differences in model competence across languages (Conneau et al., 2020; Pires et al., 2019; Rust et al., 2021). These factors mean that two translated prompts may preserve the same underlying meaning while still producing hidden states with different language-dependent structure.

This distinction matters because a probe does not simply test whether the model understands the task content. It learns a predictive relationship between hidden activations and a target property. In the case of success prediction, this means learning a direction or subspace associated with likely correctness. If this success-related structure is stable across languages, then a probe trained in one language may remain useful in another. If it shifts with the language of the prompt, the probe may fail even when the underlying model is capable of solving the translated problem. Existing work on multilingual representation learning therefore motivates the possibility of cross-lingual success prediction, but does not by itself establish whether success-related probe directions are stable across languages.

A third limitation concerns the distinction between discrimination and calibration. Many analyses of predictive signals focus on whether a model or probe can separate positive cases from negative cases. In success prediction, this means assigning higher scores to prompts that the model is likely to solve than to prompts that it is likely to fail. This form of ranking is important, and metrics such as AUROC are well suited to measuring it (Fawcett, 2006). However, ranking performance is not sufficient when predicted scores are used as probabilities. A probe may correctly rank examples while systematically overestimating or underestimating the probability of success in a target language. In that case, the probe is discriminative but not calibrated. This distinction is especially important in cross-lingual settings, where a score scale learned in one language may not transfer reliably to another (Guo et al., 2017; Jiang et al., 2022).

A final limitation is the weak connection between success decodability and practical routing. Showing that success information is present in hidden states establishes that a useful

signal may exist, but it does not show that the signal can improve deployment decisions. In cost-aware inference, the operational question is whether a system can use such a signal to decide when a cheaper model is sufficient and when a stronger model is needed. This requires more than representational evidence. The predictor must be reliable under the conditions in which it is used, and its scores must support threshold-based decisions that trade off accuracy and cost. Existing routing work has studied model selection and LLM cascades, but the connection between multilingual pre-generation probing, cross-lingual calibration, and routing remains underexplored (Chen et al., 2023; Dekoninck et al., 2025; Ong et al., 2024).

2.7.2. Thesis Positioning

This thesis addresses these gaps by studying whether pre-generation success predictors can generalise across languages. The focus is not only on whether a model can solve translated prompts, but on whether the internal information associated with likely success is represented in a sufficiently similar way across languages for a probe to transfer. This places the work at the intersection of representation probing and multilingual transfer. From the probing perspective, the thesis asks whether success is linearly accessible from hidden states. From the multilingual perspective, it asks whether this accessibility is shared across languages or tied to language-specific activation patterns.

The analysis is also explicitly layer-wise and model-wise. Since transformer models produce a sequence of hidden representations across depth, success information may not be equally accessible at every layer. Earlier layers may remain closer to token-level or surface-level features, intermediate layers may encode more shared semantic or task-level structure, and later layers may become more specialised for generation in the prompt language. Studying probes across layers therefore makes it possible to ask not only whether success information transfers, but where in the model this transfer is strongest. Comparing models of different sizes and generations further allows the analysis to examine whether transferability changes with scale and with improvements in multilingual modelling.

The evaluation is organised around three connected requirements: discrimination, calibration, and routing utility. Discrimination tests whether probes can separate likely successes from likely failures. Calibration tests whether predicted scores correspond to empirical success rates across languages. Routing then tests whether these predictions can support cost-aware model selection. This progression is important because each step adds a stronger requirement. A probe may be useful for analysing representations if it discriminates well, but a router also needs scores that can be thresholded reliably. Multilingual

routing therefore requires both cross-lingual transfer and calibrated decision signals.

The central gap addressed by the thesis is whether pre-generation success probes are reusable across languages and reliable enough to support multilingual routing. Prior work shows that success can be decoded from pre-generation activations, but it does not establish whether these success probes transfer across languages, where in model depth this transfer is strongest, or whether transferred scores remain calibrated enough for routing. Answering this requires connecting ideas that are often studied separately: hidden-state probing, cross-lingual representation transfer, calibration, and cost-aware inference. The following chapters develop this connection empirically by constructing a multilingual success-prediction setting, extracting layer-wise pre-generation representations, evaluating probe transfer across languages and models, and testing whether the resulting predictors can support routing decisions.

3 | Methodology

This chapter describes the methodology used to evaluate whether pre-generation success probes can predict model success across languages. The experimental design uses a controlled multilingual setting in which the same mathematical reasoning problems are evaluated across multiple languages, allowing the language of the prompt to vary while the underlying task remains fixed.

The methodology follows the main stages of the experiment. First, the prediction task is defined as estimating whether a model is likely to solve a mathematical reasoning problem before any answer tokens are generated. A multilingual benchmark is then constructed by translating the same MATH problems into ten languages. For each model, language, and problem, repeated generations are collected and converted into empirical success scores. These scores are paired with pre-generation hidden activations extracted from each model layer and used to train linear success probes.

The trained probes are evaluated primarily in terms of same-language prediction, cross-lingual transfer, calibration, and layer-wise behaviour. Routing is then used as a downstream evaluation of prediction quality. A reliable success probe should not only predict model behaviour in isolation, but also provide scores that can guide cost-aware decisions about whether to use a cheaper model or defer to a stronger one.

3.1. Experimental Setting

This section defines the experimental setting used throughout the thesis. It first formalises pre-generation success prediction as a model–language–problem prediction task, then describes how empirical success scores are obtained from repeated generations. It also defines the cross-lingual transfer setting and explains why the analysis is performed layer by layer across the model depth.

3.1.1. Task Definition

This thesis studies pre-generation success prediction for multilingual large language models. Let x denote a mathematical reasoning problem, ℓ the language in which the problem is presented, and m the model being evaluated. The language-specific prompt is denoted by x_ℓ. The goal is to predict whether m is likely to solve x_ℓ correctly using only the model's hidden representation of the prompt, before any response tokens are generated.

The pre-generation representation is extracted from the final token of the formatted prompt, immediately before generation begins. This position is used because, through the transformer's attention mechanism, it can attend to all earlier prompt tokens and therefore summarises the full prompt context available before the first answer token is produced. The predictor therefore operates on the model state formed after reading the prompt, but before observing the generated reasoning, final answer, or any other output tokens.

This formulation differs from standard answer evaluation. In ordinary evaluation, the model first generates a response and the response is then judged against a reference answer. In the setting considered here, the generated answer is not used as input to the predictor. Instead, the task asks whether the model's pre-generation internal state already contains information about its future success.

The prediction target is model-specific rather than problem-specific. The same mathematical problem may be solved reliably by one model, inconsistently by another, and rarely by a smaller or less capable model. Similarly, the same underlying problem may produce different outcomes when expressed in different languages. The task is therefore defined over a model–language–problem combination, rather than over the mathematical problem alone.

Mathematical reasoning problems provide a controlled setting for this analysis. The goal is not primarily to benchmark mathematical ability, but to obtain prompts for which success can be measured systematically. Since these problems usually have well-defined final answers, generated responses can be converted into correctness labels more reliably than in open-ended generation tasks. This makes the setting suitable for studying whether pre-generation hidden activations contain information about later correctness.

3.1.2. Pre-generation Success Prediction

For each model m, language ℓ, and problem x, success is estimated from repeated sampled generations. Let n denote the number of generated responses and let $k_{m,\ell,x}$ denote the

number of those responses judged correct. The empirical success score is defined as

$$y_{m,\ell,x} = \frac{k_{m,\ell,x}}{n}.$$

In this thesis, $n = 5$, so the target value belongs to the set $\{0, 0.2, 0.4, 0.6, 0.8, 1\}$. This score estimates how often model m solves problem x when the prompt is expressed in language ℓ under the generation procedure used in the experiment.

A success probe is then trained to predict this empirical success score from a pre-generation hidden representation. For a given transformer layer j, the feature vector $h^{(j)}_{m,\ell,x}$ is extracted after the model has processed the prompt but before any answer token has been generated. The probe maps this representation to a predicted success score:

$$\hat{y}^{(j)}_{m,\ell,x} = f^{(j)}(h^{(j)}_{m,\ell,x}).$$

The prediction is therefore based only on the model's internal state before generation, not on the generated answer, reasoning trace, or final output.

This formulation treats success as a probabilistic property of a model–language–problem combination. This is important because, under stochastic decoding, the same model may solve a problem in some generations and fail in others. The empirical success score therefore provides a more informative supervision signal than a single binary correctness label. It also allows the probe to model graded differences between problems that are almost always solved, almost always failed, or solved only inconsistently.

3.1.3. Cross-lingual Transfer Setting

The central experimental question is whether success probes trained on one language can generalise to prompts written in other languages. Because each translated version of a problem corresponds to the same underlying mathematical task, the benchmark makes it possible to vary the prompt language while holding the problem content fixed. This allows the analysis to test whether success-related information is represented in a form that is shared across languages, or whether the predictive signal is tied to language-specific activation patterns.

The experiments consider four main evaluation settings. The same-language setting trains and evaluates probes on held-out examples from the same language. This provides a matched-distribution reference point for how well success can be predicted when the training and test languages are identical.

The full cross-language setting trains a probe on each source language in turn and evaluates it on every different target language. With ten languages, this produces 90 ordered source–target transfer pairs for each model and layer, excluding the ten same-language pairs. This setting is used for the main cross-lingual AUROC analysis, where transfer performance is reported as the mean over all off-diagonal source–target pairs. It therefore measures whether success probes transfer across languages in general, rather than only from English to other languages.

English-trained transfer is used for calibration analysis. In this setting, the probe is trained on English examples and evaluated on each target language, including English as the in-language reference. This tests whether the score scale learned from English remains reliable when applied to other languages. The English-trained calibration results are compared with pooled multilingual probes at the same layer selected by validation negative log-likelihood.

The pooled multilingual setting trains a single probe on a language-balanced multilingual training set and evaluates it separately on each language. The probe receives only activation vectors as input and is not given language identity. This setting tests whether exposure to multilingual activation distributions during training improves robustness and calibration across languages, without giving the multilingual probe more training examples than a language-specific probe.

Strong cross-lingual transfer would indicate that success-predictive structure is at least partly shared across languages. These signals may reflect aspects of problem difficulty, model uncertainty, or language competence that are encoded in a partially shared representational space. Weak transfer, by contrast, would suggest that the success-prediction signal changes when the prompt language changes. Such shifts may arise from differences in translation quality, tokenisation, prompt length, language-specific representations, or the model's competence in a given language.

3.1.4. Layer-wise Analysis

The thesis also studies where success-related information is represented within the model. Instead of extracting activations from a single fixed layer, probes are trained independently at every available hidden-state depth. For a model with L transformer blocks, this gives $L + 1$ probed layers in total. Layer 0 corresponds to the token embedding layer, while layers $1, \ldots, L$ correspond to the outputs of transformer blocks $1, \ldots, L$. This indexing convention is used consistently across all models.

At every layer, the representation is extracted from the last token position in the prompt.

Keeping the token position fixed ensures that the layer-wise analysis captures how the pre-generation representation changes across model depth, rather than differences caused by extracting features from different positions.

Each layer is probed independently. For every source language and model, a separate probe is trained at each layer and evaluated on the validation split. For layer-wise discriminative analysis, probes are evaluated across all layers to produce full AUROC trajectories. When a single layer is required for selected-layer analyses, such as reliability diagrams, the layer is chosen using the relevant validation metric described in Section 3.5.3. This selection is performed only on validation data. Test-set metrics, such as AUROC and calibration error, are then reported at the selected layer and are not used to choose the layer.

Because the models considered in this thesis have different numbers of transformer blocks, layers are analysed using both raw layer indices and relative depth. Raw indices are used for model-specific results, while relative depth is defined as the layer index divided by L, giving a value between 0 and 1. This makes it possible to compare where success information appears across models of different depths.

The layer-wise design is important because cross-lingual transfer may depend not only on whether success information is present, but also on where it is linearly accessible. Intermediate layers may encode more abstract task-level information shared across languages, while later layers may become more closely tied to language-specific generation. Probing every layer therefore allows the experiments to test whether transferable success signals emerge at particular depths and whether same-language prediction and cross-lingual transfer rely on similar or different representational stages.

3.2. Data and Models

This section describes the benchmark and model set used in the experiments. It first explains how the multilingual MATH dataset is constructed, including the source problems, language selection, translation procedure, and data splits. It then describes the evaluated model configurations, which provide the basis for both the layer-wise probing experiments and the routing simulation. Finally, it introduces BFCL as a function-calling dataset used for the cross-dataset evaluation.

3.2.1. MATH Dataset Source

This thesis uses the MATH dataset as the source for constructing a controlled benchmark for pre-generation success prediction (Hendrycks et al., 2021). MATH is a mathematical reasoning dataset composed of competition-style problems that require more than surface-level pattern matching or short factual recall. This makes it suitable for studying model success prediction because the same model may solve some problems reliably while failing on others, creating the variation required to train and evaluate success probes.

The experimental dataset consists of 3,000 problems sampled from the full MATH dataset. The sample was selected to produce a broad range of empirical success rates in English, rather than to preserve the original distribution of MATH subject categories or difficulty annotations. Problems were then selected so that the final subset contained examples spanning different levels of empirical success, including problems that were usually solved, problems that were usually failed, and problems with intermediate success rates.

This sampling strategy reflects the purpose of the study. The probes require informative supervision across the success range. A sample concentrated mainly on very easy problems would provide few failure cases, while a sample concentrated mainly on very difficult problems would provide few success cases. Both situations would make it harder to train and evaluate success predictors. Balancing the dataset by empirical success therefore provides a more suitable benchmark for probing whether hidden activations contain information about later correctness.

Each item in the dataset contains the original English problem statement, the corresponding reference answer, and translations of the problem into the other nine languages used in the thesis. The final multilingual dataset therefore preserves the same underlying mathematical problems across languages while changing the linguistic form in which those problems are presented.

The dataset should be understood as a controlled probing benchmark rather than as a new multilingual mathematics benchmark. Its purpose is to support a systematic analysis of whether pre-generation hidden activations contain success-related information that is stable across languages.

3.2.2. Language Selection

The multilingual benchmark uses ten languages: English, Simplified Chinese, Italian, French, Swahili, Russian, Turkish, Arabic, Thai, and Telugu. This language set was chosen to provide a diverse but manageable evaluation of cross-lingual transfer in pre-

generation success probes. The aim is not to cover the full space of multilingual variation, but to include languages that differ in script, linguistic structure, tokenisation behaviour, and expected model support.

Since MATH is originally written in English, English serves as both the dataset reference language and the main source language for English-to-target transfer experiments. The remaining languages were selected to introduce variation along several dimensions. Italian and French provide Romance-language cases written in the Latin script. Russian introduces a Cyrillic-script language. Arabic, Thai, Telugu, and Simplified Chinese introduce non-Latin scripts with different segmentation and tokenisation characteristics. Turkish uses the Latin script but differs substantially from English and the Romance languages in its morphological structure. Swahili provides an additional lower-resource setting in which model competence and translation quality may be more variable.

This selection also creates variation in expected model support. Some languages in the benchmark, such as English, Chinese, French, Russian, and Arabic, are widely represented in multilingual NLP settings and are likely to be better supported by large multilingual models. Others, such as Swahili and Telugu, provide more challenging cases where lower resource availability, tokenisation fragmentation, or weaker model-language competence may affect both generation performance and success prediction. Thai and Turkish add further diversity because their difficulty for the model may arise from different factors, including segmentation behaviour in Thai and morphology in Turkish.

The purpose of this language selection is therefore not to classify languages into fixed resource categories, but to expose the probes to a range of multilingual shifts. Prior work has shown that multilingual models can benefit from shared cross-lingual representations, while still exhibiting language-specific variation in transfer behaviour (Conneau et al., 2020). Tokenisation is also relevant because multilingual tokenizers may represent different languages with different degrees of coverage and fragmentation (Limisiewicz et al., 2023). Including languages with different scripts, structures, and expected levels of model support makes it possible to evaluate whether success probes rely on language-independent signals or on representations that shift with the language of the prompt.

3.2.3. Translation Procedure

The multilingual dataset was constructed by translating the selected English MATH problems into the nine non-English languages described in Section 3.2.2. Translation was performed directly from English into each target language using DeepL Translator (DeepL SE, 2026). This direct translation setup preserves a clear relationship between each trans-

lated prompt and the original English problem, which is important for comparing model behaviour across languages.

Only the problem statement was translated. The reference answer associated with each MATH problem was kept unchanged across all language versions, since the underlying mathematical task remains the same regardless of the language used to express it. As a result, each dataset item contains the original English problem, the original reference answer, and one translated problem statement for each of the nine target languages. The dataset is stored with one row per original MATH problem, using separate columns for the English problem and each translated version.

The translation process was kept deliberately simple. Each problem statement was provided to DeepL as plain text and translated into the corresponding target language. This avoids adding extra prompt instructions that could introduce inconsistencies across languages. At the same time, mathematical problems often contain equations, symbols, and structured notation that should not be altered by translation. For this reason, additional preprocessing was applied to preserve sections enclosed in `[asy]` blocks. These blocks contain Asymptote code used to represent diagrams and graphs, and preliminary inspection showed that they were often translated incorrectly when included directly in the input text. The original English `[asy]` blocks were therefore retained in the translated prompts to preserve the intended visual content and avoid introducing diagram-level translation errors.

After translation, regular-expression checks were used to verify basic consistency properties across the translated dataset. These checks were intended to detect obvious formatting issues, such as missing protected blocks or malformed structured notation. They do not provide a complete guarantee of translation correctness, but they help ensure that the translated prompts preserve important mathematical components of the original problems.

Translation quality was also estimated automatically using `Unbabel/wmt22-cometkiwi-da`, a reference-free COMET-KIWI quality estimation model (Rei et al., 2022). For each translated problem, the model was given the English source problem and the corresponding translated problem, without using a human reference translation. Scores were computed for all translated problem statements and then averaged by language. The full translated prompt text was included in the evaluation, including preserved `[asy]` blocks. The resulting average scores are reported in Table 3.1.

The automatic quality estimates suggest that translation quality is broadly comparable across languages, while still showing some variation. Italian, French, and Turkish receive

Language	COMET-KIWI score
Italian	0.8295
French	0.8293
Turkish	0.8284
Russian	0.8139
Chinese	0.8050
Telugu	0.7934
Thai	0.7921
Swahili	0.7913
Arabic	0.7758

Table 3.1: Average reference-free COMET-KIWI quality estimation scores for the translated MATH problem statements. Scores are averaged over the 3,000 translated problems for each language.

the highest average scores, while Arabic receives the lowest average score among the evaluated languages. These scores should be interpreted as automatic quality estimates rather than definitive judgements of translation correctness. In particular, mathematical notation and language-specific wording may affect the scores in ways that do not always correspond directly to mathematical equivalence.

Automatic translation makes it possible to construct a multilingual version of the dataset at the scale required for this thesis, but it also introduces a potential source of variation. Translation errors or changes in mathematical wording may affect the apparent difficulty of some problems. For this reason, translation quality is treated as a possible limitation when interpreting cross-lingual differences in probe transfer and model success.

3.2.4. Model Selection

The experiments evaluate eight model configurations: `Qwen3-0.6B`, `Qwen3-1.7B`, `Qwen3-4B`, `Qwen3-8B`, `Qwen3.5-4B`, `Qwen3.5-9B` and `gpt-oss-20b`, evaluated both with low and high reasoning capabilities.

For compactness, the rest of the thesis refers to these configurations as evaluated models. This convention includes both independently trained checkpoints and, in the case of `gpt-oss-20b`, different reasoning effort settings of the same checkpoint referred to as `gpt-oss-low` and `gpt-oss-high`.

The Qwen-family models provide the main scale and model-family comparison for the layer-wise cross-lingual probing analysis. The Qwen3 models provide a scale comparison within one model generation, while the Qwen3.5 models provide a comparison with a

newer generation of the same broad model family. These models are selected because they are openly available, support multilingual input, span several parameter scales, and allow hidden-state extraction using standard transformer tooling. Using several models from the same broad family makes it possible to compare how success-prediction signals vary across model size, language, and layer while keeping the model family relatively consistent.

The choice of Qwen models is also motivated by their multilingual design. The Qwen3 technical report describes Qwen3 as targeting reasoning, instruction following, and multilingual capabilities, with support expanded to 119 languages and dialects compared with Qwen2.5 (Yang et al., 2025). The Qwen3.5 model cards further describe expanded global linguistic coverage, with support for 201 languages and dialects (Qwen Team, 2026). This makes the Qwen family appropriate for evaluating whether pre-generation success probes transfer across languages with different scripts, resource profiles, and linguistic structures.

The `gpt-oss-20b` configurations are included to add a reasoning effort comparison. The GPT-OSS model card describes `gpt-oss-20b` as an open-weight reasoning model based on a mixture-of-experts transformer architecture (S. Agarwal et al., 2025). In this thesis, `gpt-oss-low` and `gpt-oss-high` denote the same base model run with low and high reasoning effort, respectively. They are reported separately because the reasoning effort setting is part of the prompt configuration and leads to separate generations, empirical success labels and activation caches.

Together, these model choices support two complementary comparisons. The Qwen-family models are used to study how cross-lingual success-prediction signals vary across scale and model generation under a controlled multilingual generation setting. The `gpt-oss-20b` configurations are used to study how those signals vary across reasoning effort settings in a mixture-of-experts reasoning model.

3.2.5. Train-validation-test Splits

The dataset is split into training, validation, and test partitions by underlying MATH problem rather than by translated prompt. This ensures that different language versions of the same problem do not appear in different splits. Without this constraint, a probe could be trained on one language version of a problem and evaluated on another language version of the same problem, which would overestimate cross-lingual generalisation.

The split is deterministic and depends only on the problem identity. Each problem's base identifier is hashed using a fixed seed and the resulting value is used to assign the problem to the training, validation, or test partition according to nominal 70–15–15 thresholds.

The split is not stratified by difficulty, empirical success rate, language, subject category, or any other variable.

The same split is used for all languages and all models. That is, once a problem is assigned to a partition, all translated versions of that problem remain in the same partition for every model. The training set is used to fit probe parameters, the validation set is used to select regularisation strength and layer-level choices, and the test set is reserved for final evaluation.

3.2.6. BFCL Function-calling Dataset

The thesis also evaluates the Berkeley Function-Calling Leaderboard (BFCL) (Patil et al., 2025). BFCL evaluates whether a model can produce a correct structured function call for a given natural-language request and function specification. This differs from MATH, where the model is evaluated on mathematical reasoning. BFCL is then evaluated separately from the multilingual transfer setting, since it concerns a different task format rather than a different prompt language.

The BFCL data used in this thesis include ten subsets, combining non-live v3 and live v4 data. The non-live subsets are simple_python, multiple, parallel, and parallel_multiple. The live subsets are live_simple, live_multiple, live_parallel, live_parallel_multiple, live_irrelevance, and live_relevance. Together, these subsets cover different function-calling conditions, including single-call, multiple-call, parallel-call, and relevance-based cases.

3.3. Generation Procedure

This section describes how model responses are generated and converted into empirical success labels. It first defines the multilingual prompt format used across models, then specifies the sampling protocol used to obtain repeated responses. It finally explains how final answers are extracted, normalised, and aggregated into the success scores used as probe targets.

3.3.1. Prompt Construction

Prompts are formatted as two-message chats consisting of a system message and a user message. The system message provides the general instruction for solving the problem, while the user message contains the translated mathematical problem followed by a short

language-specific suffix. The general structure is:

system:	`<system_prompt>`
user:	`<problem_text><user_suffix>`
assistant:	generation starts here.

Both the system prompt and the user suffix are written in the same language as the problem statement. This keeps the instruction context aligned with the language of the input problem, rather than using an English instruction for every language. For example, in English the system message is:

```
"Please reason step by step, and put your final answer within \boxed{}."
```

and the user suffix is:

```
"Let's think step by step and output the final answer within \boxed{}."
```

The same structure is used for all other languages, with translated system prompts and user suffixes. The instruction to place the final answer inside `\boxed{}` appears both in the system message and in the suffix appended to the problem. This redundancy is intentional: it encourages the model to produce a step-by-step solution while making the final answer easier to extract during correctness labelling. The full set of language-specific system prompts and user suffixes is reported in Appendix A.

The final prompt is serialised using each model's own chat template through `apply_chat_template`. This ensures that the same two-message structure is converted into the format expected by each evaluated model, while allowing model-specific control options to be applied where necessary.

For Qwen-family models, explicit thinking mode is disabled by setting `enable_thinking = False`. Although the prompts still ask the model to reason step by step in the visible response, disabling explicit thinking mode is important for the multilingual design of the experiment. Preliminary testing showed that, when thinking mode was enabled, the models often produced reasoning traces in English even when the input problem was written in another language. Since this thesis studies success prediction from multilingual prompt representations, a systematic shift toward English reasoning would reduce the role of the prompt language. Disabling explicit thinking mode therefore helps preserve the multilingual nature of the experiment while still allowing ordinary step-by-step reasoning in the generated response.

For `gpt-oss-20b`, the same base checkpoint is used for both evaluated settings. The differ-

ence between `gpt-oss-low` and `gpt-oss-high` is introduced by adding a reasoning-effort instruction to the system message before serialisation with the model's chat template. In particular, the script prepends the following line to the language-specific system instruction:

```
gpt-oss-low:   Reasoning: low
gpt-oss-high:  Reasoning: high
```

The reasoning-control line is followed by the normal language-specific system instruction. Conceptually, the two GPT-OSS settings therefore receive the same translated task prompt, but with a different reasoning-effort instruction prepended to the system message.

For BFCL, prompts are constructed using the benchmark's function-calling format rather than the multilingual MATH prompt template. The official BFCL system prompt and function definitions are retained, and all BFCL prompts are evaluated in English. For the Qwen models, response generation uses BFCL's manually defined Qwen formatting. The expected output is BFCL's Python-style abstract syntax tree format, such as `[function_name(arg=value), ...]`, rather than native XML or JSON tool-call syntax.

3.3.2. Answer Generation Protocol

For the MATH benchmark, five sampled responses are generated for each model, language, and problem. These repeated generations are used to estimate the model's empirical success rate on each model–language–problem combination. The MATH generation parameters are kept consistent across evaluated models: temperature is set to 0.7, top-p is set to 0.8, and $n = 5$ responses are generated for each prompt. The same translated problem set and correctness-labelling pipeline are used for all evaluated models.

Model responses are generated using vLLM. Hidden-state extraction is performed separately using Hugging Face Transformers, as described in Section 3.4. This separation is used because vLLM supports efficient batched generation for collecting large-scale rollouts, while Hugging Face Transformers provides direct access to the layer-wise hidden states required for probing.

The maximum generation length is set according to the model context window:

$$\texttt{max\_tokens} = 32768 - 2500 - 500,$$

where 32768 is the maximum context length, 2500 tokens are reserved as a prompt budget, and 500 tokens are kept as a safety buffer. This setting gives the models enough room to produce complete mathematical solutions while reducing the risk of exceeding the context

window.

The same broad generation protocol is used for the Qwen-family models and for both `gpt-oss-20b` reasoning-effort regimes. The main model-specific difference is the prompt format described in Section 3.3.1: Qwen models are evaluated with explicit thinking mode disabled, while `gpt-oss-low` and `gpt-oss-high` are evaluated using their corresponding reasoning-effort settings. Apart from these model-specific formatting and reasoning-mode differences, the underlying translated problem statements, sampling parameters, number of rollouts, maximum generation length, and correctness evaluation are kept consistent.

For BFCL, responses are generated only for the six Qwen-family models: `Qwen3-0.6B`, `Qwen3-1.7B`, `Qwen3-4B`, `Qwen3-8B`, `Qwen3.5-4B`, and `Qwen3.5-9B`. The `gpt-oss-20b` reasoning-effort settings are not included in the BFCL experiment. Each BFCL prompt is sampled five times, using temperature 0.7 and top-$p = 0.95$.

3.3.3. Correctness Labelling

Each generated response is converted into a binary correctness label by comparing the model's final answer with the reference answer from MATH. The labelling procedure is fully automatic and rule-based.

The prompt instructs the model to place its final answer inside `\boxed{}`. For this reason, answer extraction first identifies the content of the last `\boxed{}` expression in the generated response. A brace-depth parser is used so that nested braces inside the boxed answer are handled correctly. If no boxed answer is found, the response is marked incorrect. No additional fallback is used to extract the last mathematical expression or infer an answer from the reasoning trace.

The extracted prediction and the reference answer are then passed through the same normalisation pipeline. This normalisation removes superficial formatting differences while preserving the mathematical content needed for comparison. Outer math and brace delimiters are stripped, selected LaTeX formatting commands are standardised or removed, Unicode minus signs are replaced with ASCII hyphens, and whitespace is removed. Unit annotations, percent signs, dollar signs, and thousand-separator commas are removed when applicable. Numeric strings are also normalised by stripping leading zeros from integers and trailing `.0` from decimals. Multi-value answers are standardised by converting textual conjunctions such as `\text{and}` into comma separators, single-letter multiple-choice answers are uppercased, and outer parentheses or brackets are stripped.

Correctness is then checked using a sequence of rule-based comparisons. First, the nor-

malised prediction is compared with the normalised reference answer using exact string equality. When the reference contains an equality chain, such as $120 \times 4 = 480$, the right-hand side is also considered as a valid normalised form. Second, if both the prediction and reference can be parsed as numeric values, they are compared by numeric equivalence with a relative tolerance of 10^{-9}. This numeric parser supports plain decimals, fractions written as `\frac{a}{b}`, slash notation such as a/b, and ratio notation such as $a : b$. Coordinate pairs are compared element-wise. Third, simple single-variable inequalities are converted to interval notation and compared against interval-form references, and the reverse conversion is also supported.

Only the final extracted answer is evaluated. A response with correct reasoning but no boxed final answer is marked incorrect, while formatting variation in a boxed answer can still be accepted if it is handled by the normalisation and equivalence rules described above. The output of this procedure is a binary label for each sampled response, which is later aggregated across repeated generations to compute the empirical success score.

BFCL responses are labelled using the benchmark-specific evaluation procedure. Instead of extracting a boxed mathematical answer, the evaluator checks whether the generated Python-style function call matches the expected function call for the prompt. The output of this procedure is again a binary correctness label for each sampled response, but the parsing and equivalence rules are specific to the BFCL function-calling format rather than to mathematical answer matching.

3.3.4. Empirical Success Scores

After correctness labelling, the binary labels for the sampled responses are aggregated into an empirical success score for each model–language–problem combination. Let $k_{m,\ell,x}$ denote the number of correct responses and let $n_{m,\ell,x}$ denote the number of generated responses available for that combination. The empirical success score is defined as

$$y_{m,\ell,x} = \frac{k_{m,\ell,x}}{n_{m,\ell,x}}.$$

In the normal case, five responses are generated for each problem, so $n_{m,\ell,x} = 5$ and the target value belongs to the set $\{0, 0.2, 0.4, 0.6, 0.8, 1\}$.

This score estimates the model's probability of solving the problem under the specific generation configuration used in the experiment. It should therefore be interpreted as an empirical success rate averaged over five independent samples drawn with the sampling parameters described in Section 3.3.2. It is not an intrinsic measure of problem difficulty

independent of the model, language, or decoding setup.

Truncated generations are not automatically removed from the dataset. If a truncated response still contains a boxed final answer, it is scored using the same correctness procedure as any other response. A rollout is therefore judged by the extracted final answer when one is available, rather than being discarded solely because it reached the maximum generation length.

For evaluation metrics that require binary targets, the empirical success score is binarised using a threshold of 0.5. Problems with $y_{m,\ell,x} > 0.5$ are treated as positive examples, meaning that the model tends to solve the problem under the rollout configuration. Problems with $y_{m,\ell,x} \leq 0.5$ are treated as negative examples. Probe predictions are clipped to the interval $[0, 1]$ at evaluation time, but the empirical targets themselves are not clipped or smoothed.

3.4. Representation Extraction

This section describes how pre-generation hidden activations are extracted from the formatted prompts. The same extraction principle is used across the MATH and BFCL settings: the model processes the prompt, no answer token is generated, and hidden states are collected at the point where generation would normally begin.

3.4.1. Extraction Objective

The purpose of representation extraction is to construct the feature vectors used by the success probes. For each model m, language ℓ, problem x_i, and layer j, one pre-generation activation vector is extracted and paired with the empirical success score defined in Section 3.3.4.

Each feature vector is extracted from the formatted prompt before any answer tokens are generated. The representation is therefore independent of the sampled rollouts used to compute the success score. In other words, the model produces stochastic generations during rollout collection, but only a single pre-generation representation is extracted from the prompt itself. This separation is essential because the probe is intended to test whether the model's internal state before answering contains information predictive of future correctness.

No generated text is included in the extraction input. If generated answer tokens were included, the probe could exploit information from the model's own response rather than measuring what is represented before the response begins. The extraction procedure

therefore captures the model state at the point of generation onset, after the prompt has been encoded but before the first answer token is produced.

3.4.2. Prompt Encoding and Forward Pass

Hidden-state extraction is performed using Hugging Face Transformers. The prompts used for extraction are constructed in the same way as the prompts used during answer generation. For MATH, each prompt contains the system instruction, the translated problem statement, and the language-specific user suffix described in Section 3.3.1. The model's chat template is then applied with a generation prompt, so that the input ends with the assistant-turn prefix where answer generation would normally begin.

For BFCL, the extraction input uses the same prompt produced by the BFCL handler for response generation. This ensures that the hidden states correspond to the exact function-calling prompt format evaluated in the experiment, including the benchmark system prompt and function definitions.

The extraction step consists of a single forward pass over the formatted prompt. No autoregressive decoding is performed and no answer tokens are produced. The model is run in evaluation mode with gradients disabled, and hidden states are returned for all layers. This produces the layer-wise internal representations of the prompt that are later used for probing.

For `gpt-oss-20b`, extraction is performed separately for the low and high reasoning effort regimes. Since the reasoning-effort setting is included in the system prompt, the two regimes correspond to different formatted prompts before generation begins. The resulting activation caches are therefore matched to the same inference regimes used to generate the empirical success labels.

Unlike rollout generation, representation extraction is deterministic for a fixed model and formatted prompt. There is no sampling or decoding during this step, so the resulting activation cache can be reused across probe training, validation, testing, and downstream analysis without recomputing hidden states.

3.4.3. Final-token Hidden State

For each layer, the feature vector is extracted from the final valid token of the formatted prompt. This is the last token of the assistant-turn prefix appended by the chat template. The input therefore ends at the point where the model is about to begin generating its answer, but no answer token has yet been produced.

The use of the final prompt position follows the task definition in Section 3.1.1. In a decoder-only transformer, this position can attend to all earlier prompt tokens through causal attention and therefore summarises the full prompt context available before generation begins. Keeping this token position fixed across all layers ensures that differences between extracted feature vectors reflect changes across model depth, rather than differences in the token position being analysed.

3.4.4. Layer-wise Feature Construction

Activations are extracted for every available hidden-state layer. For a model with L transformer blocks, this produces $L + 1$ layers. Layer 0 corresponds to the embedding output before any transformer block, while layers $1, \ldots, L$ correspond to the residual-stream outputs after transformer blocks $1, \ldots, L$. This convention matches the layer indexing introduced in Section 3.1.4.

For each layer j, the final-token hidden state is extracted as a vector

$$\mathbf{h}_{m,\ell,i}^{(j)} \in \mathbb{R}^{d_m},$$

where d_m is the hidden dimension of model m. The hidden dimension differs across models, but this does not require any cross-model feature alignment because probes are trained separately for each model.

The raw activations are cached to disk before probe training. For each model–language pair, the cache stores the layer-wise activation matrices and the corresponding problem identifiers. Success scores and split assignments are loaded separately from the per-problem dataset files and matched to the activation rows by problem identifier during probe training.

No centering, standardisation, normalisation, or dimensionality reduction is applied before caching. The stored activations therefore represent the raw final-token hidden states produced by the model. Any feature standardisation used by the probe is applied later during probe training and is fitted only on the training split.

3.5. Probe Training

This section describes the linear probes used to predict empirical success from pre-generation activations.

3.5.1. Probe Objective

The purpose of probe training is to test whether model success is linearly decodable from pre-generation hidden activations. For each model, language, and layer, a probe is trained to map the activation vector extracted in Section 3.4 to the empirical success score defined in Section 3.3.4. The target is the raw empirical success rate $y \in \{0, 0.2, 0.4, 0.6, 0.8, 1\}$, which is used directly without smoothing or transformation. For BFCL, the same objective is applied to English function-calling prompts, without iterating over target languages.

The probe outputs a continuous predicted success score $\hat{y}$, interpreted as an estimate of the model's empirical success rate for the given prompt under the generation procedure used in the experiment. This regression formulation preserves the graded nature of the supervision signal. For example, a problem solved in one out of five samples and a problem solved in all five samples both provide evidence about future success, but they should not be treated as equally successful cases.

The probe is deliberately restricted to the pre-generation representation itself. It receives the final-token activation vector from a single layer and is not given any additional information about the prompt, the language, the tokenised length, or the generated response. This ensures that probe performance reflects information recoverable from the model's internal state before generation, rather than from external metadata or from the answer produced afterwards.

Probes are trained separately for each model because the hidden dimension differs across models. They are also trained separately for each layer, allowing the analysis to identify where success-related information is most accessible across model depth.

The probes should be interpreted as linear decodability tests rather than causal explanations of model behaviour. A successful probe shows that success-related information is recoverable from the hidden representation by a simple linear model, but it does not establish that the language model itself uses the same direction or feature during generation. At the same time, because the probe operates on a single pre-generation activation vector and does not require autoregressive decoding, it is lightweight compared with full answer generation. This makes it suitable for the routing experiments described later, where the success score is used before deciding whether to use a cheaper model or defer to a stronger one.

3.5.2. Ridge Regression Formulation

Given a hidden activation vector from layer j,

$$\mathbf{h}_{m,\ell,i}^{(j)} \in \mathbb{R}^{d_m},$$

the ridge probe predicts the empirical success score as

$$\hat{y}_{m,\ell,i}^{(j)} = \mathbf{w}^\top \tilde{\mathbf{h}}_{m,\ell,i}^{(j)} + b,$$

where $\tilde{\mathbf{h}}_{m,\ell,i}^{(j)}$ is the standardised activation vector, $\mathbf{w}$ is the learned weight vector, and b is an intercept term. For a fixed model, layer, and training configuration, the probe is trained by minimising

$$\sum_i \left(y_{m,\ell,i} - \hat{y}_{m,\ell,i}^{(j)} \right)^2 + \alpha \|\mathbf{w}\|_2^2,$$

where α controls the strength of the L_2 regularisation penalty.

Before fitting the ridge probe, activation features are standardised using statistics computed only from the probe's training split. For each model, layer, and training condition, a `StandardScaler` is fitted on the training activations and then reused unchanged for validation and test examples. This prevents information from the validation or test splits from influencing the feature normalisation. The empirical success target y is not standardised or transformed, since it directly represents the observed success rate under the generation procedure.

Ridge regression is used because the extracted activation vectors are high-dimensional, while each language-specific training split contains only a few thousand examples. Without regularisation, a linear probe could overfit to incidental directions in activation space. The L_2 penalty discourages large weights and provides a more stable estimate of whether success information is linearly accessible from the representation.

The ridge objective does not constrain predictions to lie inside the empirical success range. As a result, the raw probe output may occasionally fall below 0 or above 1, even though the target is an empirical success score. For downstream evaluation, probe predictions are therefore clipped to the interval $[0, 1]$. This clipping is applied only after prediction and does not change the fitted probe parameters or the training objective.

3.5.3. Regularisation Selection

The regularisation coefficient α is selected using validation-set performance. The grid of candidate values is $\alpha \in \{1, 10, 100, 1000, 10000, 100000, 1000000\}$ and the selection is performed independently for each source language, model, and layer.

The validation metric depends on the purpose of the experiment. For layer-wise discriminative analysis, probes are evaluated across all layers using AUROC on the test set after regularisation has been selected on validation data. These full trajectories are used to study where success information is most accessible across model depth. For calibration experiments, the reported probes are evaluated at the layer selected by validation negative log-likelihood. This allows the reliability diagrams and ECE summaries to compare score calibration at a fixed selected layer.

When negative log-likelihood is used, the empirical success scores are first converted into binary labels using the same threshold adopted for AUROC evaluation: examples with $y > 0.5$ are treated as successful, while examples with $y \leq 0.5$ are treated as unsuccessful. Probe predictions are clipped to $[0, 1]$ before computing negative log-likelihood.

Regularisation and layer selection are performed using validation data only. For each layer, the probe is trained with every candidate value of α, and the value with the best validation score is selected for that layer. When a single selected layer is required, the best layer is then chosen according to the relevant validation metric. Test-set metrics are computed only after these choices have been made.

3.5.4. Training Configurations

The MATH experiments use two main probe-training configurations: language-specific training and pooled multilingual training. These configurations determine which examples are used to fit the probe. Cross-lingual transfer is then evaluated by applying language-specific probes to target languages different from the training language.

In the language-specific configuration, a separate probe is trained for each model, language, and layer using only the training split for that language. This produces one probe per model–language–layer combination. This configuration is used both as a matched-language reference point and as the source of probes for cross-lingual transfer analyses.

In the cross-lingual transfer setting, the trained source-language probe is applied unchanged to target-language activations. For example, an English-trained probe is evaluated directly on the activations extracted from other target languages, without fitting

any part of the probe pipeline on those target languages. This preserves the zero-shot transfer setting, since neither the probe parameters nor the associated training statistics are estimated from the target-language evaluation data.

In the pooled multilingual configuration, a single probe is trained for each model and layer using a language-balanced multilingual training set. The pooled training set has the same total number of examples as a language-specific training split. It is formed by sampling equal-sized subsets from the ten languages, so that each language contributes 10% of the multilingual training examples.

The pooled multilingual probe receives only activation vectors as input and is not given the language identity. Its performance therefore reflects whether a single linear mapping from pre-generation activations to empirical success can be learned across the multilingual activation space, rather than whether the probe can use an explicit language indicator to adjust its predictions.

For the BFCL configuration, a separate probe is trained for each Qwen-family model and layer using only BFCL training examples. Since the BFCL data used in this thesis are English-only, this configuration does not include a language-specific training dimension. The same feature standardisation, ridge-regression objective and regularisation procedure are used as in the MATH experiments.

For the joint MATH+BFCL configuration, training examples from both datasets are pooled without subsampling. The ridge objective is fitted with sample weights so that MATH examples collectively contribute 50% of the total training loss and BFCL examples collectively contribute the remaining 50%, regardless of the number of examples in each dataset. The probe receives only activation vectors as input and is not given an explicit dataset or task indicator.

3.6. Routing Simulation

This section defines the routing simulation used in the thesis. It first specifies the model-selection objective, then introduces the routing policies, the encoder-target decoupled variant, and the cost model used to compare routed systems.

3.6.1. Routing Objective

The routing experiment evaluates whether pre-generation success probes can support cost-aware model selection. For each problem–language example, the system must choose one model from a set of candidate models that differ in expected success and generation cost.

The objective is to reduce the cost of always using the strongest model while preserving as much of its expected success as possible.

The candidate model set is

`Qwen3-0.6B`	`Qwen3-1.7B`
`Qwen3-4B`	`Qwen3-8B`
`Qwen3.5-4B`	`Qwen3.5-9B`
`gpt-oss-low`	`gpt-oss-high`

The model `gpt-oss-high` is used as the anchor model. It represents the strongest reference model in the evaluated set and defines the baseline against which routed success and routed cost are compared.

For each candidate model $m \in \mathcal{M}$, language ℓ, and problem x, the router uses a predicted success score

$$\widehat{y}_{m,\ell,x}.$$

This score estimates the empirical success rate that would be obtained if model m were selected for that problem–language example. The router combines these predicted scores with the cost model defined below to select one target model for each example. The resulting routed system is then evaluated by comparing its expected success and expected cost with the anchor baseline.

This setup turns success prediction into a model-selection problem. A probe is useful for routing only if its scores identify cases where a cheaper model is likely to be sufficient, while still deferring harder or uncertain examples to stronger models. The routing experiment therefore tests whether the calibration and transfer properties measured in the probing analysis translate into practical accuracy–cost trade-offs.

3.6.2. Routing Policies

The routing experiment compares two reusable probe configurations. The English-only transfer router uses probes trained on English examples and applies their predicted success scores to all target languages. This setting tests whether a success signal learned from English activations is sufficient for multilingual model selection. The pooled multilingual router uses the multilingual probes described in Section 3.5.4, which are trained on pooled examples and evaluated across target languages. This setting tests whether multilingual training produces a score scale that is more suitable for routing.

For each probe configuration, routing is evaluated using two decision rules. The first is a

utility-based router. For each candidate model m, the router computes

$$U_{m,\ell,x} = \widehat{y}_{m,\ell,x} - \lambda c_m,$$

where $\widehat{y}_{m,\ell,x}$ is the predicted empirical success score for model m on problem x in language ℓ, c_m is the fixed expected generation cost of model m, and λ is a cost-penalty parameter. The selected model is the one with the highest utility:

$$r(\ell, x) = \arg\max_{m \in \mathcal{M}} U_{m,\ell,x}.$$

Small values of λ prioritise predicted success and therefore tend to select stronger models. Larger values of λ increase the penalty assigned to expensive models and therefore make the router more willing to select cheaper models.

The second decision rule is an anchor-aware router. This router uses the same utility scores, but gives the anchor model `gpt-oss-high` a conservative safeguard. Let

$$m^* = \arg\max_{m \in \mathcal{M}} U_{m,\ell,x}$$

be the model with the highest utility score, and let a denote the anchor model. The router selects the cheaper utility-optimal model only when its utility exceeds the anchor utility by more than a tolerance parameter δ:

$$r(\ell, x) = \begin{cases} m^*, & \text{if } U_{m^*,\ell,x} > U_{a,\ell,x} + \delta, \\ a, & \text{otherwise.} \end{cases}$$

This rule makes routing more conservative. When the utility advantage of a cheaper model is small, the router keeps the anchor rather than switching away from the strongest model. The tolerance parameter therefore controls how much evidence is required before the router moves away from the anchor.

The utility-based router and the anchor-aware router represent two different operating regimes. The utility-based router directly explores the trade-off between predicted success and cost, and can produce aggressive cost savings when λ is large. The anchor-aware router instead treats the strongest model as the default choice and routes away from it only when the probe scores suggest that a cheaper model is sufficiently competitive. This makes the anchor-aware rule more conservative and better suited to evaluating whether probe-based routing can reduce cost without sacrificing success.

The parameters λ and δ are selected using validation data. The resulting routing policies

are then evaluated on the held-out test split. Reporting the full range of validation-selected operating points makes it possible to compare the accuracy–cost trade-off induced by English-only transfer and pooled multilingual training.

3.6.3. Encoder-target Decoupling

The routing policies described in section 3.6.2 use success scores predicted from the activations of each candidate target model. This is a natural way to use model-specific probes, but it requires the prompt to be processed by every candidate model before the routing decision is made. In cost terms, this means that direct probe-based routing pays the input or prefill cost for all candidate models, even though only one model is ultimately selected to generate the final answer.

To reduce this prefill overhead, the thesis also considers an encoder-target decoupled routing setting. In this setting, a single encoder model processes the prompt and provides the hidden representation used to predict the success of all candidate target models. The encoder is therefore separated from the target model selected for generation. This follows the idea of encoder-target decoupling (ETD), where the model used to compute routing features is not necessarily the model used to answer the prompt (Varshney et al., 2026).

Let e denote the encoder model and m denote a candidate target model. For each encoder, target model, layer, and training setting, a separate ridge probe is trained to predict the empirical success of target model m from the pre-generation activation of encoder e. The predicted score $\widehat{y}_{m,\ell,x}$ represents the estimated success of target model m on problem x in language ℓ. The role of the encoder is to provide the representation from which this score is predicted.

The ETD probes are independent target-specific heads rather than a single predictor shared across target models. For a fixed encoder and layer, one ridge probe is trained for each candidate target model. The target set remains the full routing candidate set, including both `gpt-oss-low` and `gpt-oss-high`.

The ETD probes use the same training procedure described in Section 3.5. Activations are extracted from the final prompt token, standardised with a `StandardScaler`, and used to train ridge-regression probes. Predictions are clipped to the interval $[0, 1]$. The regularisation coefficient is selected independently for each encoder, target model, layer, and training setting using validation AUROC.

Once the predicted scores $\widehat{y}_{m,\ell,x}$ have been computed, the routing decision is exactly the same as in Section 3.6.2. Encoder-target decoupling therefore does not introduce a new

routing rule. It only changes the cost of obtaining the predicted success scores, because the router can use a single encoder prefill instead of separate prefills from all candidate target models.

As a secondary variant of encoder-target decoupled routing, the thesis also evaluates layer ensembles. In this setting, multiple layer-specific ridge probes are trained for the same encoder and target-model prediction task. Each probe produces a predicted success score from a different encoder layer, and the final routing score is obtained by averaging these predictions. The averaged score is then used by the same routing policy as the single-layer ETD score. This ensemble is applied at the score level rather than by concatenating hidden representations. It does not require an additional encoder forward pass, since the activations for all layers are obtained during the same prompt prefill.

3.6.4. Cost Model

Each routing policy is assigned an expected inference cost. The cost is designed to capture the main token-level expenses incurred by pre-generation routing: prompt processing and answer generation. Prompt processing corresponds to the input or prefill stage, where a model reads the prompt and constructs the internal state from which generation can begin. Answer generation corresponds to the output stage, where the selected model produces response tokens.

The cost model therefore combines a per-token price, which reflects model scale, with per-example token counts, which reflect the amount of prompt processing and generation performed for each example. This follows the same general principle used by Dekoninck et al. (2025) in their routing and cascading experiments, where inference cost is approximated from token count and model size. Costs are measured relative to the anchor baseline, which always uses `gpt-oss-high`.

The simulation distinguishes between two uses of cost. The routing decision uses the fixed model-level proxy c_m that is based on the average output length of each model on the training split and is available before routing decisions are made. By contrast, the final evaluation cost is computed after a model has been selected, using the prompt-prefill costs actually incurred by the routing strategy and the recorded output-tokens of the selected model.

Each model is first assigned a price per million output tokens. For dense models, the output price is proportional to the number of parameters:

$$C_{\mathrm{m}}^{\mathrm{out}} = 0.10 P_m,$$

where P_m is the number of parameters of model m, measured in billions. For example, `Qwen3-4B` is assigned a price of \$0.40 per million output tokens, while `Qwen3.5-9B` is assigned a price of \$0.90 per million output tokens.

The GPT-OSS models are mixture-of-experts models. In a mixture-of-experts architecture, the model contains a larger set of total parameters, but only a subset of these parameters is activated for each token. The active parameters therefore approximate the part of the model that contributes most directly to per-token computation, while the total parameters still matter because they affect memory requirements and the infrastructure needed to host the model. For this reason, GPT-OSS pricing is not based only on total parameter count. Instead, it assigns separate weights to active and total parameters:

$$C^{\text{out}}_{\text{gpt-oss}} = 0.090 P_{\text{active}} + 0.045 P_{\text{total}}.$$

Using $P_{\text{active}} = 3.6$ and $P_{\text{total}} = 20$, this gives

$$C^{\text{out}}_{\text{gpt-oss}} = 0.090 \times 3.6 + 0.045 \times 20 = 1.224.$$

Both `gpt-oss-low` and `gpt-oss-high` are therefore assigned an output price of \$1.224 per million output tokens.

Input tokens are priced as a fixed fraction of output tokens:

$$C^{\text{in}}_m = \frac{1}{6} C^{\text{out}}_m.$$

This is a modelling assumption motivated by current commercial API pricing, where several flagship models price input tokens at one sixth of output tokens, including OpenAI's `gpt-5.5` and multiple Google Gemini models such as Gemini 3.5 Flash and Gemini 3.1 Pro (Google, 2026; OpenAI, 2026). Anthropic's Claude pricing provides a slightly more conservative comparison, with several models pricing input tokens at approximately one fifth of output tokens (Anthropic, 2026).

Table 3.2 reports the output model costs used in the routing simulation. Although `gpt-oss-low` and `gpt-oss-high` have the same per-token price, their expected generation costs differ because `gpt-oss-high` produces substantially longer outputs on average.

This cost model is a simplified token-level proxy rather than a full deployment cost model. It includes prompt-prefill and output-generation costs through input and output token prices, but it does not account for probe overhead, batching, hardware utilisation or latency differences.

Model	Input price ($/M tok.)	Output price ($/M tok.)	Avg. Tokens	Exp. cost ($/k gen.)	Relative Exp. cost
Qwen3-0.6B	0.010	0.060	588.60	0.0353	0.0107
Qwen3-1.7B	0.028	0.170	2300.24	0.3910	0.1181
Qwen3-4B	0.067	0.400	2067.44	0.8270	0.2497
gpt-oss-low	0.204	1.224	947.16	1.1593	0.3501
Qwen3.5-4B	0.067	0.400	3007.59	1.2030	0.3633
Qwen3-8B	0.133	0.800	1587.36	1.2699	0.3835
Qwen3.5-9B	0.150	0.900	2487.29	2.2386	0.6760
gpt-oss-high	0.204	1.224	2705.42	3.3114	1.0000

Table 3.2: Model generation costs used in the routing simulation. Input and output prices are reported per million tokens. The expected cost is computed from the output price and the average output length on the training split. Relative cost is measured with respect to always selecting `gpt-oss-high`.

3.7. Evaluation

This section describes the evaluation procedures used to assess base model performance, probe discrimination, calibration, cross-lingual transfer, and routing behaviour.

3.7.1. Base Model Performance

Probe performance is interpreted alongside the empirical success rates of the evaluated models. These success rates describe the behaviour that the probes are trained to predict and provide context for differences across models and languages. In particular, they indicate how often each model solves the benchmark problems under the generation procedure used in the experiment, before any probing model is considered.

For each model and language, the base success rate is computed by averaging the empirical success scores over the test split. This produces a model–language summary of observed correctness. These values are not probe metrics, since they do not evaluate the quality of the learned predictor. Instead, they describe the target distribution against which probe predictions are assessed.

Reporting base success rates is important because the difficulty of the probing task depends partly on the distribution of successes and failures. If a model solves very few problems in a given language, the positive class becomes small; if a model solves most problems, the evaluation contains relatively few failure cases. In the multilingual setting,

these differences are especially relevant because they may reflect not only problem difficulty, but also language-specific model competence, tokenisation effects, or translation-related variation.

3.7.2. Discriminative Evaluation

Discriminative evaluation measures whether the probe assigns higher scores to examples that the model is more likely to solve than to examples it is more likely to fail.

Although probes are trained as regression models on the raw empirical success scores, AUROC requires a binary target. The empirical success score is therefore binarised using the threshold introduced in Section 3.3.4. Examples with $y > 0.5$ are treated as positive, while examples with $y \leq 0.5$ are treated as negative. Since each empirical success score is computed from five sampled generations, this corresponds to treating a problem as successful when the model solves it in at least three out of five generations.

The main discriminative metric is AUROC. AUROC measures whether positive examples tend to receive higher predicted scores than negative examples across possible decision thresholds. It is therefore appropriate for evaluating whether success-related information is linearly recoverable from the hidden representation, without committing to a fixed threshold for deployment or routing.

For the layer-wise discriminative results, AUROC is reported across all probed layers. These trajectories are used descriptively to analyse where same-language and cross-lingual success signals become accessible. Peak AUROC values are then used to summarise the strongest discriminative performance across layers.

3.7.3. Calibration Evaluation

Calibration evaluation examines whether predicted success scores have a reliable probabilistic interpretation. A calibrated probe should produce scores that match observed empirical success rates among examples assigned similar predictions. For example, if a group of examples receives predicted scores close to 0.7, then the mean empirical success rate of that group should also be close to 0.7.

Calibration is evaluated using both reliability diagrams and Expected Calibration Error (ECE). Reliability diagrams provide a visual comparison between predicted success scores and observed empirical success rates. ECE provides a scalar summary of the average calibration error across prediction bins.

To construct a reliability diagram, clipped probe predictions are divided into ten equal-width bins over the interval $[0, 1]$. Bins that contain fewer than five predictions are skipped. For each bin, the x-axis value is the mean predicted score in that bin, while the y-axis value is the mean raw empirical success score of the same examples. The empirical success score is not binarised for this analysis, so the diagram directly compares predicted success scores with the graded empirical targets used for probe training.

A perfectly calibrated probe would lie close to the diagonal line, where predicted success equals observed empirical success. Deviations from this diagonal indicate miscalibration. Curves below the diagonal indicate overestimation, while curves above the diagonal indicate underestimation. This distinction is important because a probe can rank examples well while still producing scores that are unreliable as probability estimates.

3.7.4. Layer-wise Evaluation

Layer-wise evaluation measures how success-prediction performance changes across model depth. Since a separate ridge probe is trained for each layer, each model produces an AUROC trajectory rather than a single probe score. These trajectories are used to identify where success information is most accessible and whether the same depth patterns appear under same-language prediction and cross-lingual transfer.

The full trajectories show where success-related information becomes linearly accessible and whether performance increases, plateaus, or declines as the prompt representation is transformed through the model. This is especially important for comparing same-language prediction with cross-lingual transfer, since transferable success signals may be strongest at different depths from same-language signals.

For calibration experiments, a single layer is selected using validation negative log-likelihood. This selected layer is used to construct the reliability diagrams and ECE summaries. The test set is not used during either regularisation selection or layer selection.

This evaluation design separates two questions. The full trajectories show how success information is distributed across model depth, while selected-layer calibration results evaluate whether the resulting scores are reliable as probability estimates.

3.7.5. Cross-lingual Transfer Evaluation

Cross-lingual transfer evaluation measures whether success probes remain useful when the language of the prompt changes. The central comparison is between probes evaluated under same-language conditions and probes evaluated under language shift. This makes it

possible to distinguish success information that is recoverable only within a single language from success information that is encoded in a form that transfers across languages.

Language-specific probes provide the main reference point. For each model and language, a probe is trained on the training split of that language and evaluated on held-out examples from the same language. These results indicate how well success can be predicted when the probe is trained and tested on the same prompt distribution.

Full cross-language transfer is evaluated by training probes on each source language and evaluating the trained probes unchanged on every different target language. Same-language pairs are excluded. With ten languages, this produces 90 ordered source–target pairs per model and layer. The main transfer AUROC reported in the Results chapter is the mean over these off-diagonal pairs.

English-trained transfer is used separately for the calibration comparison. In that setting, the probe is trained only on English examples and evaluated unchanged on all ten target languages, including English as the in-language reference. This provides a single-source transfer setting in which calibration can be compared across target languages using the same learned score scale.

The pooled multilingual setting provides the main comparison for multilingual calibration. In this setting, the probe is trained on the language-balanced multilingual training set described in Section 3.5.4 and then evaluated separately on each target language. The probe receives only activation vectors and is not given language identity. This tests whether balanced exposure to multilingual variation during training improves the reliability of the learned score scale across target languages.

3.7.6. Routing Evaluation

Routing performance is evaluated in terms of expected success and expected inference cost. For each problem–language example, the router selects one candidate model according to the policy defined in Section 3.6.2. The success assigned to that decision is the empirical success score of the selected model on the same example. Thus, if $r(\ell, x)$ denotes the model selected for problem x in language ℓ, the expected routed success is

$$\text{Success}_{\text{router}} = \frac{1}{N} \sum_{(\ell,x)} y_{r(\ell,x),\ell,x},$$

where N is the number of evaluated problem–language examples and $y_{r(\ell,x),\ell,x}$ is the empirical success score computed from five sampled generations. This use of empirical

success rates gives an estimate of expected correctness under the same generation protocol used to construct the probe labels.

The expected inference cost of a routing policy is computed using the token-level prices defined in Section 3.6.4. Since direct probe-based routing uses a separate probe for each model, the prompt must be processed by each candidate model before the routing decision is made. The evaluated router cost therefore includes the prompt-prefill cost of all candidate models, plus the expected output-generation cost of the selected model:

$$\mathrm{Cost}_{\mathrm{router}} = \frac{1}{N} \sum_{(\ell,x)} \left[\sum_{m \in \mathcal{M}} T^{\mathrm{in}}_{\ell,x,m} \frac{C^{\mathrm{in}}_m}{10^6} + \bar{T}^{\mathrm{out}}_{\ell,x,r(\ell,x)} \frac{C^{\mathrm{out}}_{r(\ell,x)}}{10^6} \right].$$

Here $T^{\mathrm{in}}_{\ell,x,m}$ is the number of input tokens required to encode problem x in language ℓ with model m, and $\bar{T}^{\mathrm{out}}_{\ell,x,r(\ell,x)}$ is the mean output length of the selected model on that example, estimated from the repeated generations.

Savings are reported relative to the anchor baseline. Let a denote the anchor model, `gpt-oss-high`. The anchor cost is the expected cost of always selecting a, including its prompt-prefill cost and expected output-generation cost:

$$\mathrm{Cost}_{\mathrm{anchor}} = \frac{1}{N} \sum_{(\ell,x)} \left[T^{\mathrm{in}}_{\ell,x,a} \frac{C^{\mathrm{in}}_a}{10^6} + \bar{T}^{\mathrm{out}}_{\ell,x,a} \frac{C^{\mathrm{out}}_a}{10^6} \right].$$

The router saving is then

$$\mathrm{Saving}_{\mathrm{router}} = 1 - \frac{\mathrm{Cost}_{\mathrm{router}}}{\mathrm{Cost}_{\mathrm{anchor}}}.$$

A saving of zero corresponds to the cost of always using the anchor model. Positive values indicate that the router reduces expected inference cost, while negative values indicate that the prompt-prefill cost required to compute the routing scores outweighs the savings obtained by selecting cheaper models.

Encoder-target decoupling is evaluated with the same success and savings metrics, but with a different input-cost term. Instead of processing the prompt with every candidate model, ETD uses a single encoder model to compute the routing features. When the selected target model differs from the encoder, the target model must also process the prompt before generation. ETD therefore reduces the input-cost component from all candidate prefills to one prefill when the encoder is selected, or two prefills when a different target model is selected.

Accuracy differences are reported in percentage points relative to the anchor baseline:

$$\Delta\text{Success} = \text{Success}_{\text{router}} - \text{Success}_{\text{anchor}}.$$

Routing is first evaluated across model depth. For each relative layer depth, the routing parameters are selected on the validation split and the resulting policy is evaluated on the test split. This analysis measures whether routing performance depends on the layer used for probing, and identifies the layers used in the main comparison between English-trained transfer probes and pooled multilingual probes.

The main routing result is the accuracy–cost trade-off curve computed on the held-out test split at the selected layers. For each probe configuration, the routing parameters are varied to produce a set of policies with different levels of expected success and cost saving. In the utility-based router, this is done by sweeping the cost penalty λ. In the anchor-aware router, both the cost penalty λ and the anchor tolerance δ are varied. The resulting curve shows the range of trade-offs supported by each probe configuration, rather than reducing routing behaviour to a single threshold.

In addition to the full trade-off curve, selected operating points are highlighted for interpretation. These points are chosen from the test-set trade-off curve to represent qualitatively different regimes. In particular, the results report the highest-success policy and the policy that most closely matches the anchor model's expected success while reducing cost. These selected points are not used to tune the probes themselves; they are reported to make the trade-off curve easier to interpret and to compare the behaviour of English-only transfer and pooled multilingual routing.

For each selected operating point, the routing mix is also reported. The routing mix gives the fraction of examples assigned to each candidate model. This makes it possible to interpret the aggregate success and cost values by showing whether a policy mainly relies on the anchor model, frequently selects intermediate models, or routes aggressively to cheaper models. When combined with the accuracy–cost curve, the routing mix helps describe whether cost reductions come from selective replacement of the anchor or from broader routing toward cheaper models.

The central comparison is whether pooled multilingual training improves the accuracy–cost trade-off over English transfer, while preserving a reusable routing setting that does not require separate language-specific probes.

4 | Results

This chapter reports the empirical results, moving from base model performance to layer-wise probing, calibration, and multilingual routing.

4.1. Empirical Success on MATH

Before evaluating the success probes, it is necessary to describe the behaviour that the probes are trained to predict. The target variable in this thesis is not an intrinsic property of a mathematical problem alone, but the empirical success rate of a particular model when solving that problem in a particular language. A problem may therefore have different target values depending on the model being evaluated and the language in which the prompt is expressed. Reporting base model performance provides context for the later probing results, because probe difficulty depends on the distribution of successes and failures in each model–language setting.

Table 4.1 reports the empirical success rates of the evaluated models across the ten languages. Each value is computed by averaging the empirical success scores over the test split. Since each empirical success score is estimated from five sampled generations, the table reflects the probability that a model solves a problem under the generation procedure used in this thesis, rather than the outcome of a single deterministic generation.

The results show a clear improvement with scale within the Qwen3 family. Mean success increases from 0.148 for Qwen3-0.6B to 0.595 for Qwen3-8B. The Qwen3.5 models are substantially stronger, with Qwen3.5-4B reaching a mean success rate of 0.746 and Qwen3.5-9B reaching 0.814. GPT-OSS with low reasoning effort performs similarly to Qwen3.5-4B on average, while GPT-OSS with high reasoning effort achieves the strongest overall mean success rate, 0.859.

Performance also varies substantially across languages. English is the easiest language on average, with a mean success rate of 0.738, followed by Chinese and French, both with mean success rates of 0.661. Russian and Italian also remain among the stronger languages, while Swahili and Telugu are the most difficult languages overall, with mean

Language	Qwen3 0.6B	Qwen3 1.7B	Qwen3 4B	Qwen3 8B	Qwen3.5 4B	Qwen3.5 9B	gpt-oss low	gpt-oss high	Mean
Arabic	0.134	0.312	0.537	0.600	0.785	0.841	0.749	0.885	0.605
Chinese	0.220	0.408	0.630	0.638	0.851	0.898	0.754	0.886	0.661
English	0.279	0.508	0.716	0.728	0.916	0.935	0.895	0.925	0.738
French	0.179	0.388	0.613	0.663	0.848	0.895	0.789	0.911	0.661
Italian	0.160	0.378	0.621	0.654	0.835	0.880	0.755	0.897	0.648
Russian	0.170	0.384	0.611	0.661	0.837	0.889	0.761	0.884	0.650
Swahili	0.034	0.124	0.145	0.418	0.536	0.683	0.587	0.647	0.397
Telugu	0.056	0.153	0.322	0.435	0.497	0.580	0.716	0.849	0.451
Thai	0.138	0.324	0.526	0.578	0.590	0.715	0.695	0.826	0.549
Turkish	0.107	0.282	0.526	0.574	0.760	0.821	0.754	0.882	0.588
Mean	0.148	0.326	0.525	0.595	0.746	0.814	0.746	0.859	0.595

Table 4.1: Empirical success rates of the evaluated models across languages. Each value is the average empirical success score over the test split, where each problem-level score is computed from five sampled generations. The final row reports the average success rate for each model across languages, and the final column reports the average success rate for each language across models.

success rates of 0.397 and 0.451 respectively. This variation confirms that the probing task is not uniform across languages: the same mathematical problems lead to different observed success distributions depending on prompt language and evaluated model.

The comparison between GPT-OSS low and GPT-OSS high also shows that reasoning effort changes the empirical target distribution. High reasoning effort improves success in every language, with particularly large gains for Arabic, Telugu, Thai, and Turkish. This supports treating the two GPT-OSS settings as separate evaluated models in the probing analysis, even though they are based on the same underlying base model.

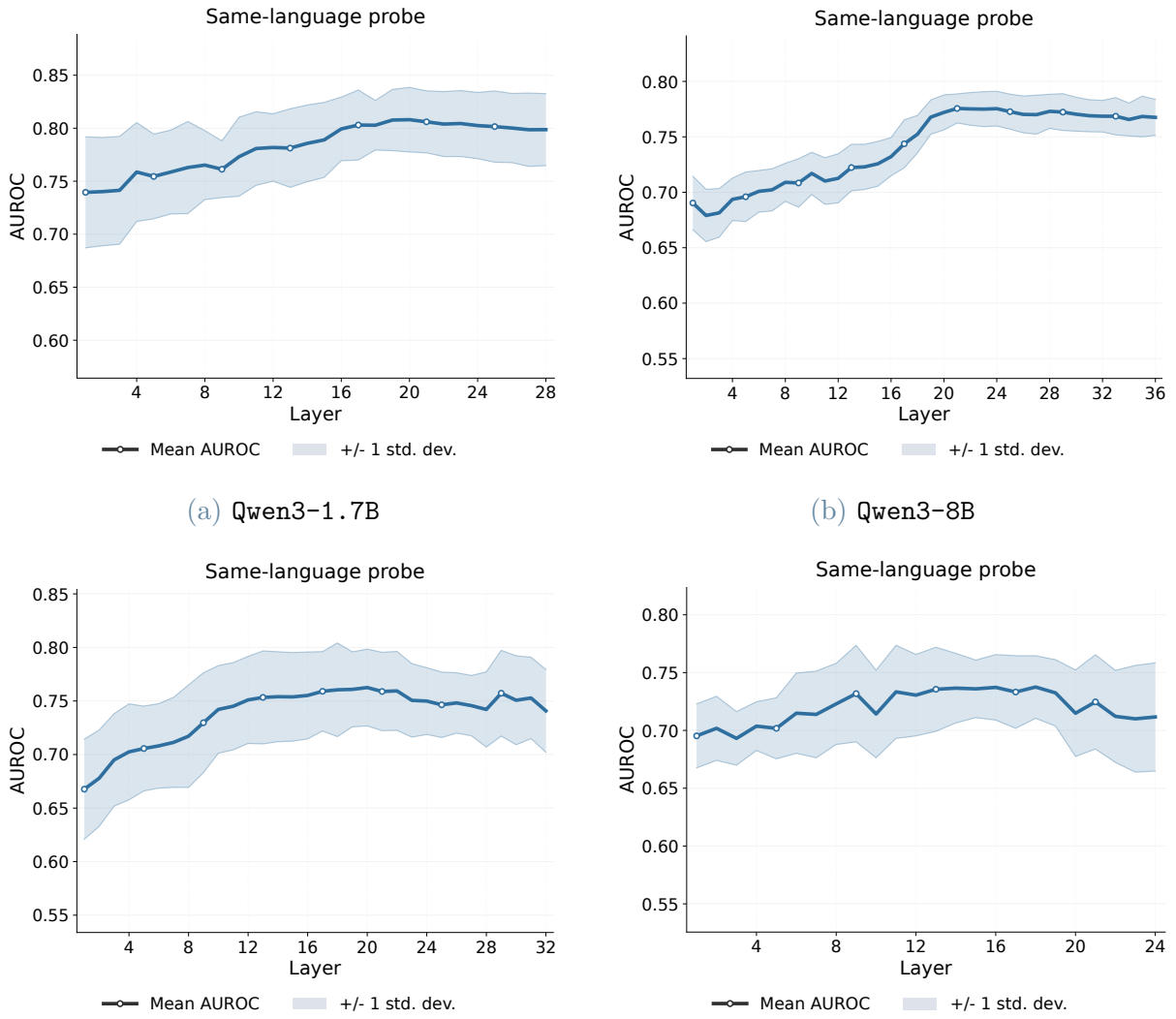
4.2. Layer-wise Success Prediction

This section analyses how success-prediction performance changes across model depth, first within languages and then under cross-lingual transfer.

4.2.1. Same-language Prediction

Same-language probes show that model success is linearly decodable from pre-generation activations across all evaluated models. In this setting, each probe is trained and evaluated on the same language, using held-out examples from the same problem-level split. The

main quantity of interest is the mean same-language AUROC across the ten languages at each layer, which provides a within-language reference point before considering cross-lingual transfer.



(a) `Qwen3-1.7B` (b) `Qwen3-8B`

(c) `Qwen3.5-9B` (d) `gpt-oss-high`

Figure 4.1: Representative layer-wise same-language AUROC trajectories. Each panel reports the mean same-language AUROC across the ten languages, with shaded variation across languages. The selected models cover Qwen3 1.7B and 8B, Qwen3.5 9B, and the strongest GPT-OSS reasoning effort setting. Same-language prediction generally improves through the early layers and remains close to its peak through the middle-to-late layers.

Figure 4.1 shows representative layer-wise same-language AUROC trajectories. Across the selected models, same-language prediction improves from the earlier layers and then reaches a broad plateau in the middle-to-late part of the network. This pattern is visible in the mean curves, which rise through the early layers and then remain relatively

stable rather than dropping sharply near the final layers. The full set of same-language trajectories for all evaluated models is reported in Appendix B.

Table 4.2 summarises the peak of the mean same-language AUROC trajectory for each evaluated model. Peak same-language AUROC ranges from 0.738 for `gpt-oss-high` to 0.823 for `Qwen3-0.6B`. All evaluated models therefore achieve substantially above-random discrimination, indicating that pre-generation hidden activations contain information that separates likely successes from likely failures before any answer tokens are generated. The corresponding language-level peak AUROC values are reported in Appendix B.1.

Model	Peak layer	Depth	Peak AUROC
`Qwen3-0.6B`	19/28	67.9%	0.823
`Qwen3-1.7B`	20/28	71.4%	0.808
`Qwen3-4B`	25/36	69.4%	0.774
`Qwen3-8B`	21/36	58.3%	0.776
`Qwen3.5-4B`	18/32	56.2%	0.795
`Qwen3.5-9B`	20/32	62.5%	0.763
`gpt-oss-low`	17/24	70.8%	0.744
`gpt-oss-high`	18/24	75.0%	0.738

Table 4.2: Peak same-language AUROC for each evaluated model. Values report the maximum of the mean same-language AUROC trajectory across layers, where the mean is computed over the ten language-specific same-language probes. Depth reports the peak layer as a percentage of the final probed layer.

The peak locations show that same-language prediction is strongest in middle-to-late layers for all evaluated models. The peak of the mean trajectory occurs between 56.2% and 75.0% of model depth. The Qwen models peak between 56.2% and 71.4% of depth, while the two GPT-OSS settings peak later, at 70.8% for `gpt-oss-low` and 75.0% for `gpt-oss-high`. Compared with the earlier unstandardised results, the main pattern remains the same: same-language AUROC rises through the early layers and remains strong in later layers.

Same-language AUROC is strong across all evaluated models, but it does not increase monotonically with model scale or base model success rate. The highest peak mean AUROC is obtained by `Qwen3-0.6B`, even though it has the lowest average base success rate in Table 4.1. The GPT-OSS settings, despite having high empirical success rates, have the lowest peak same-language AUROC values in Table 4.2. These results show that same-language probe AUROC and base model success rate describe different aspects of model behaviour.

4.2.2. Cross-lingual Transfer

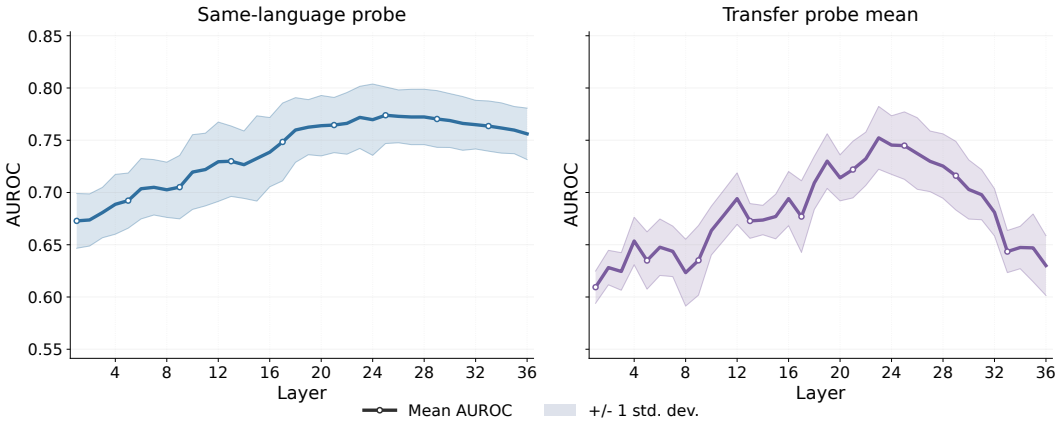
Cross-lingual transfer evaluates whether probes trained on one language remain discriminative when applied to another language. In this setting, each source-language probe is evaluated on target-language activations without refitting any part of the probe pipeline on the target language. Transfer AUROC is computed over all ordered source–target language pairs where the source and target languages differ, and then averaged to obtain a layer-wise transfer trajectory for each evaluated model.

Figure 4.2 shows representative same-language and cross-lingual transfer trajectories for `Qwen3-4B`, `Qwen3.5-4B`, and `Qwen3-8B`. The left panel of each subfigure reports same-language AUROC, while the right panel reports cross-lingual transfer AUROC. The selected models illustrate the main layer-wise transfer patterns across the evaluated models. The full set of transfer trajectories for all evaluated models is reported in Appendix B.

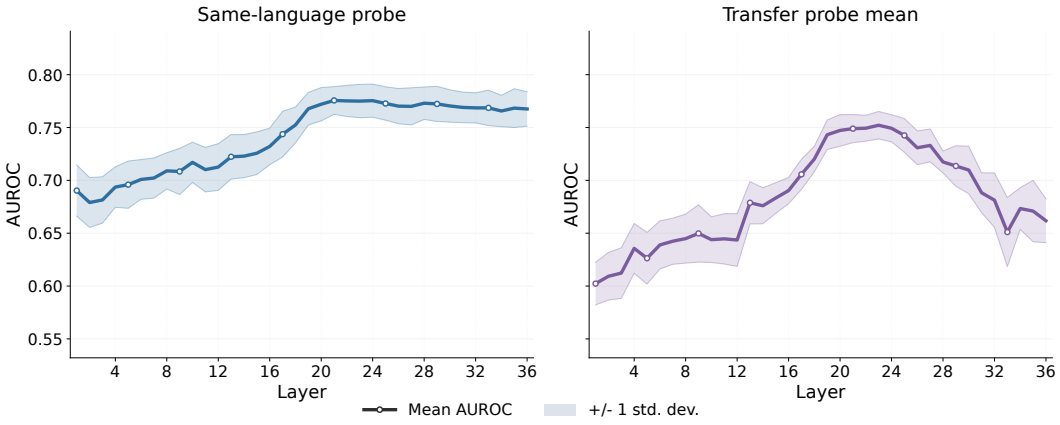
The trajectories show that cross-lingual transfer follows the same broad early-layer increase as same-language prediction, but it is more sensitive to depth after reaching its peak. This pattern is clear in both `Qwen3-4B` and `Qwen3-8B`. In these models, transfer AUROC increases through the early and middle layers, peaks in the middle-to-late part of the network, and then declines visibly toward the final layer. The `Qwen3.5-4B` trajectory follows the same general structure, but the decline after the peak is more limited. This indicates that the depth-dependent transfer pattern is shared across the selected models, while the strength of the final-layer decline varies across model families and scales.

Model	Peak layer	Depth	Peak AUROC	Final AUROC	Drop
`Qwen3-0.6B`	19/28	67.9%	0.785	0.704	0.080 (10.2%)
`Qwen3-1.7B`	17/28	60.7%	0.776	0.667	0.109 (14.0%)
`Qwen3-4B`	23/36	63.9%	0.752	0.630	0.122 (16.2%)
`Qwen3-8B`	23/36	63.9%	0.752	0.662	0.090 (12.0%)
`Qwen3.5-4B`	20/32	62.5%	0.757	0.706	0.051 (6.7%)
`Qwen3.5-9B`	19/32	59.4%	0.720	0.670	0.049 (6.8%)
`gpt-oss-low`	17/24	70.8%	0.690	0.643	0.047 (6.8%)
`gpt-oss-high`	17/24	70.8%	0.673	0.608	0.066 (9.8%)

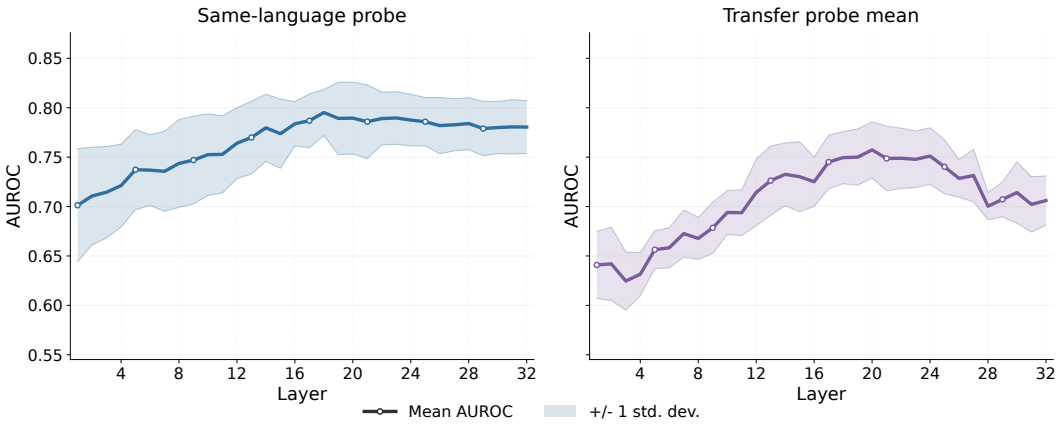
Table 4.3: Peak cross-lingual transfer AUROC for each evaluated model. Transfer AUROC is computed as the mean over all ordered source–target language pairs where the source and target languages differ. Final AUROC is measured at the final probed layer, and the drop reports the absolute and relative decrease from peak to final-layer transfer performance.



(a) `Qwen3-4B`



(b) `Qwen3-8B`



(c) `Qwen3.5-4B`

Figure 4.2: Representative layer-wise same-language and cross-lingual transfer AUROC trajectories. In each subfigure, the left panel shows same-language AUROC and the right panel shows transfer AUROC. Curves report mean AUROC with shaded variation across languages.

Table 4.3 reports the peak cross-lingual transfer AUROC for all evaluated models. Peak transfer AUROC ranges from 0.673 for `gpt-oss-high` to 0.785 for `Qwen3-0.6B`. Across all models, the peak occurs between 59.4% and 70.8% of model depth, again placing the strongest transfer performance in middle-to-late layers.

Table 4.4 compares same-language and transfer AUROC at each model's transfer-optimal layer. The gap is small but consistent across all evaluated models. It ranges from 0.020 for `Qwen3-4B` to 0.060 for `gpt-oss-high`. This shows that cross-lingual transfer retains much of the same-language discriminative signal at the transfer-optimal layer, although transfer AUROC remains lower than same-language AUROC in every model.

Model	Layer	Same-language AUROC	Transfer AUROC	Gap
`Qwen3-0.6B`	19	0.823	0.785	0.039 (4.7%)
`Qwen3-1.7B`	17	0.803	0.776	0.026 (3.2%)
`Qwen3-4B`	23	0.772	0.752	0.020 (2.6%)
`Qwen3-8B`	23	0.775	0.752	0.023 (3.0%)
`Qwen3.5-4B`	20	0.789	0.757	0.032 (4.1%)
`Qwen3.5-9B`	19	0.761	0.720	0.041 (5.4%)
`gpt-oss-low`	17	0.744	0.690	0.055 (7.4%)
`gpt-oss-high`	17	0.733	0.673	0.060 (8.2%)

Table 4.4: Same-language and cross-lingual transfer AUROC at the transfer-optimal layer for each evaluated model. The gap reports the absolute and relative decrease from same-language AUROC to transfer AUROC at that layer.

Overall, the transfer results show that cross-lingual success prediction is discriminatively effective but weaker than same-language prediction. The strongest transfer performance occurs in middle-to-late layers, while the final layer is consistently below the transfer peak. This establishes the layer-wise transfer pattern used in the following calibration analysis, where the question is whether transferred probe scores remain numerically reliable across target languages.

4.3. Calibration of Success Probes

The previous results evaluate whether success probes can discriminate between likely successes and failures. This section considers whether the predicted scores are also numerically reliable across languages. This distinction is important because AUROC measures ranking quality, whereas routing policies and threshold-based decisions require predicted scores whose numerical values remain meaningful across target languages.

4.3.1. Calibration Under English-only Transfer

The first calibration experiment considers the English transfer setting. A probe is trained on English and then evaluated separately on all ten target languages by comparing predicted success scores with empirical success rates. The English panel provides the in-language reference, while the remaining panels show how the same score scale behaves under cross-lingual transfer. Each reliability diagram is shown at a fixed model-specific layer, matching the layer used in the figure annotations. The full set of English-trained reliability diagrams for all evaluated models is reported in Appendix B.3.1.

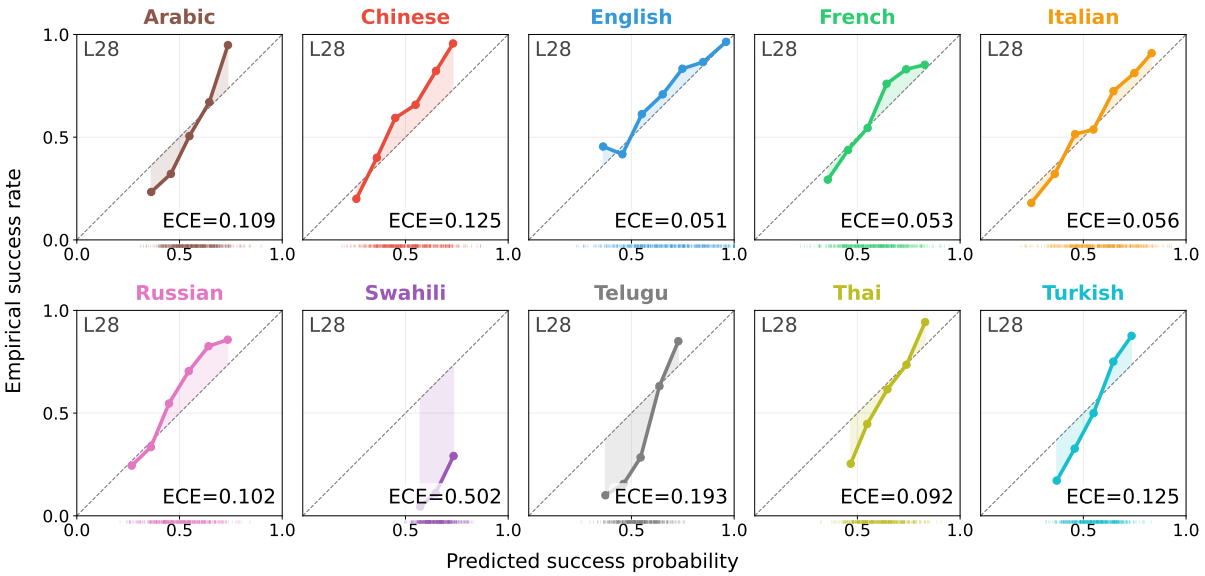


Figure 4.3: Reliability diagrams for the English-trained transfer probe on `Qwen3-4B`. The probe is trained on English and evaluated separately on each target language at the fixed layer shown in the panels. English provides the in-language reference, while the other panels show calibration under cross-lingual transfer.

Figures 4.3 and 4.4 show that English-only transfer can produce substantial calibration shifts across target languages. For `Qwen3-4B`, the English panel has an ECE of 0.051, while the worst transferred target is Swahili with an ECE of 0.502. For `Qwen3.5-4B`, the English panel is very well calibrated, with an ECE of 0.010, but transferred calibration is much less stable, with Swahili reaching an ECE of 0.470. This shows that a low English in-language calibration error does not guarantee that the same score scale remains reliable when applied to other languages.

Table 4.5 reports the mean ECE of the English-trained probe for all evaluated models. The mean ECE is computed over the ten target-language panels shown in the reliability diagrams. The table also reports the English in-language ECE, the range of non-English

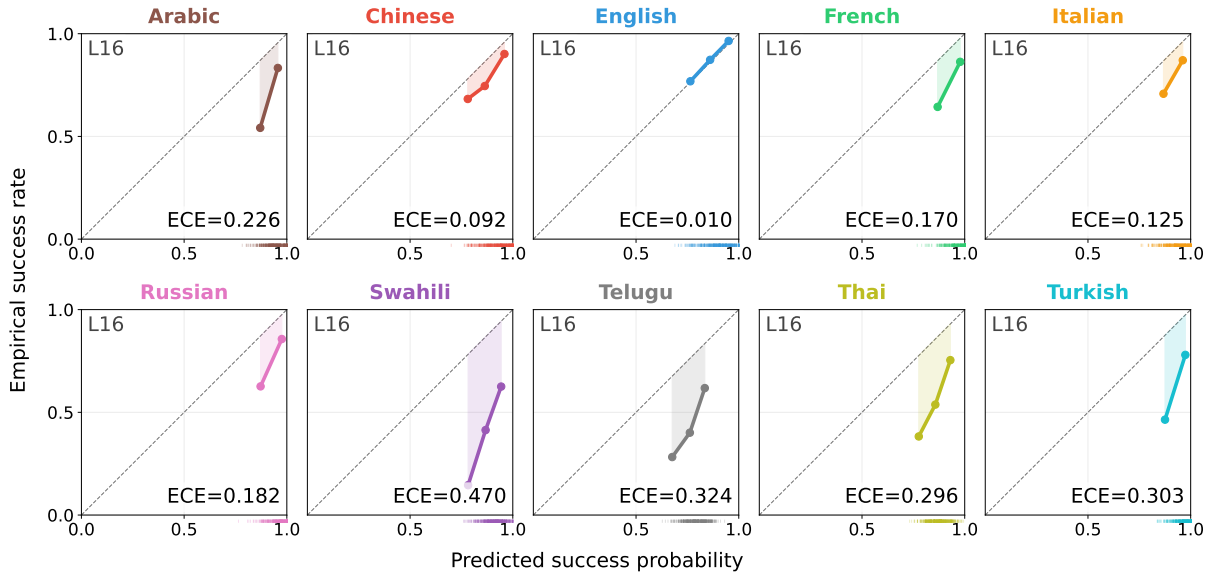


Figure 4.4: Reliability diagrams for the English-trained transfer probe on `Qwen3.5-4B`. The probe is trained on English and evaluated separately on each target language at the fixed layer shown in the panels.

Model	Layer	Mean ECE	English ECE	Non-English ECE range	Worst target
`Qwen3-0.6B`	19	0.252	0.049	0.024–0.589	Swahili
`Qwen3-1.7B`	18	0.140	0.044	0.062–0.370	Telugu
`Qwen3-4B`	28	0.141	0.051	0.053–0.502	Swahili
`Qwen3-8B`	24	0.102	0.066	0.084–0.156	Telugu
`Qwen3.5-4B`	16	0.220	0.010	0.092–0.470	Swahili
`Qwen3.5-9B`	15	0.079	0.034	0.035–0.150	Telugu
`gpt-oss-low`	9	0.141	0.043	0.085–0.250	Swahili
`gpt-oss-high`	9	0.090	0.046	0.029–0.158	Swahili

Table 4.5: Calibration error under English-only transfer. The English-trained probe is evaluated on each target language at the fixed model-specific layer shown in the reliability diagrams. Mean ECE is averaged over all ten target-language panels. The non-English ECE range excludes the English panel.

ECE values, and the target language with the highest calibration error.

The table shows that English in-language calibration can be much better than the worst transferred calibration, but that this advantage is not sufficient to make the score scale reliable across languages. For every evaluated model, the English ECE is below the highest non-English ECE. The largest target-language errors occur for Swahili with `Qwen3-0.6B`, `Qwen3-4B`, `Qwen3.5-4B`, `gpt-oss-low`, and `gpt-oss-high`, and for Telugu with `Qwen3-`

1.7B, Qwen3-8B, and Qwen3.5-9B. These results show that English-trained probes can transfer discriminatively while still producing language-dependent calibration errors. The next subsection evaluates whether pooled multilingual training produces a more consistent score scale across target languages.

4.3.2. Calibration with Pooled Multilingual Training

Pooled multilingual training evaluates whether a single probe trained with multilingual supervision produces a more reliable score scale across target languages. The probe is trained using the pooled multilingual configuration described in Section 3.5.4 and evaluated separately on each target language. Unlike the English-only transfer setting, this experiment is not a zero-shot transfer test, since each evaluated language contributes to the pooled training set. For each model, the pooled multilingual reliability diagram is reported at the NLL-selected best layer.

Figures 4.5 and 4.6 show the same two representative models used in the English-only calibration analysis. This allows the English-trained and pooled multilingual settings to be compared directly for Qwen3-4B and Qwen3.5-4B. The full set of pooled multilingual reliability diagrams for all evaluated models is reported in Appendix B.3.

The reliability diagrams show that pooled multilingual training reduces the calibration

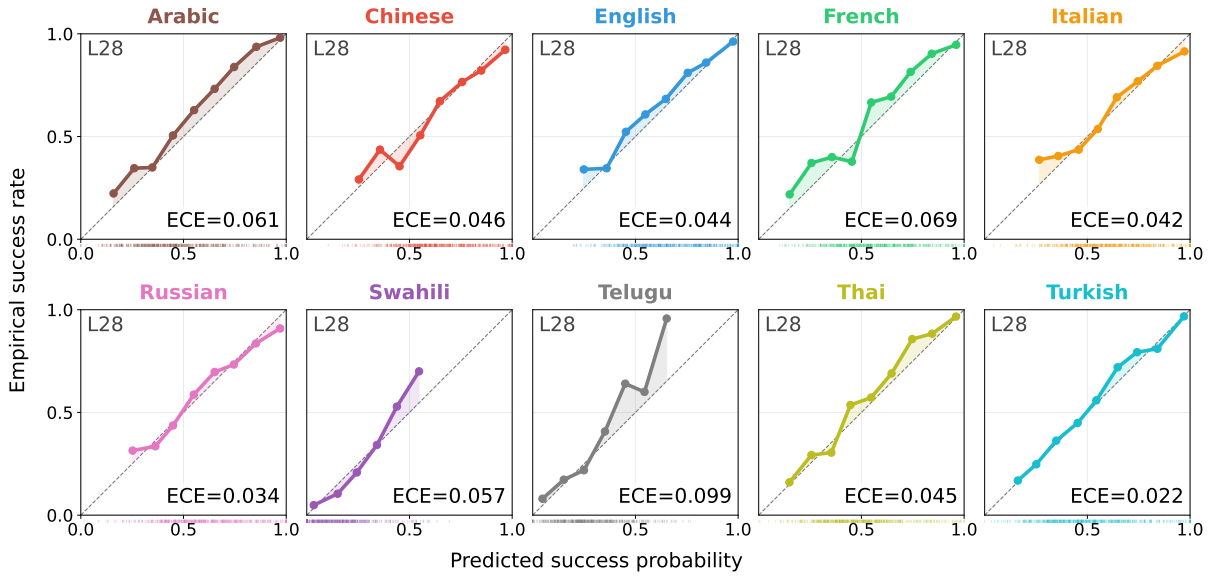


Figure 4.5: Reliability diagrams for the pooled multilingual probe on Qwen3-4B. The probe is evaluated separately on each target language at the NLL-selected best layer. Compared with the English-trained transfer probe, the pooled multilingual probe produces a more consistent score scale across target languages.

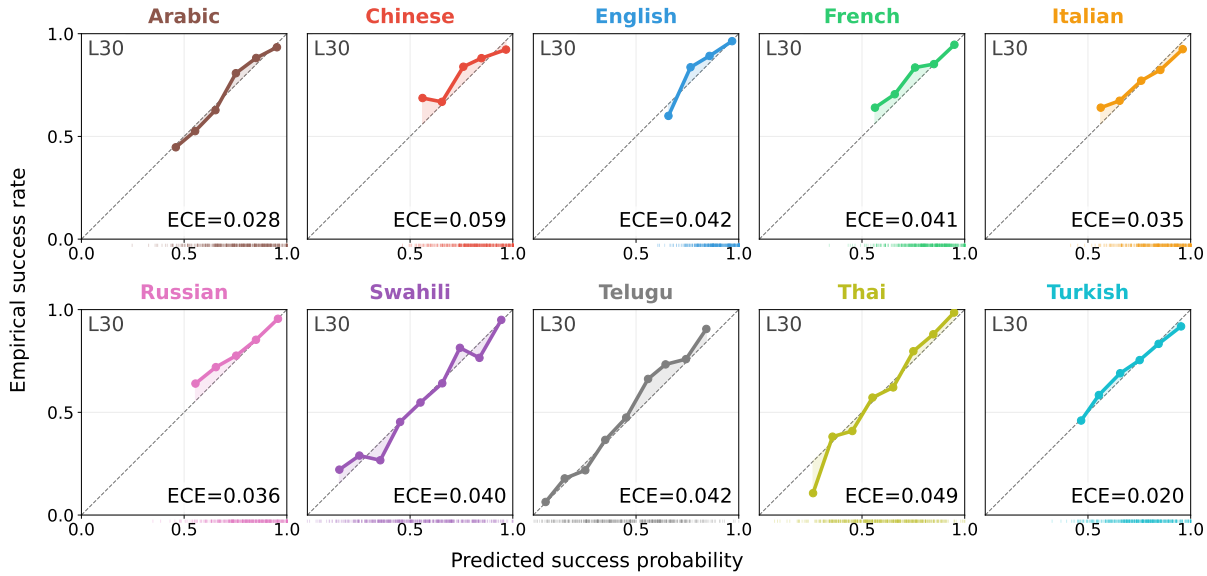


Figure 4.6: Reliability diagrams for the pooled multilingual probe on `Qwen3.5-4B`. The probe is evaluated separately on each target language at the NLL-selected best layer. The pooled multilingual setting reduces the target-language calibration shifts observed under English-only transfer.

shifts observed under English-only transfer. For `Qwen3-4B`, the mean ECE decreases from 0.141 for the English-trained probe to 0.052 for the pooled multilingual probe. For `Qwen3.5-4B`, the decrease is larger, from 0.220 to 0.039. In both cases, the pooled multilingual probe produces reliability curves that are more consistently aligned with the diagonal across target languages.

Model	English-probe ECE	Pooled ECE	Improvement
`Qwen3-0.6B`	0.252	0.067	73% reduction, 3.76× lower
`Qwen3-1.7B`	0.140	0.070	50% reduction, 2.00× lower
`Qwen3-4B`	0.141	0.052	63% reduction, 2.71× lower
`Qwen3-8B`	0.102	0.053	48% reduction, 1.93× lower
`Qwen3.5-4B`	0.220	0.039	82% reduction, 5.61× lower
`Qwen3.5-9B`	0.079	0.045	42% reduction, 1.74× lower
`gpt-oss-low`	0.141	0.039	72% reduction, 3.59× lower
`gpt-oss-high`	0.090	0.032	64% reduction, 2.79× lower

Table 4.6: Mean calibration error for English-trained and pooled multilingual probes. ECE is averaged over the ten target languages for each evaluated model. The improvement column reports both the relative reduction in mean ECE and the multiplicative decrease obtained by pooled multilingual training.

Table 4.6 summarises this comparison for all evaluated models. Pooled multilingual training reduces mean ECE for every model. The relative reduction ranges from 42% for `Qwen3.5-9B` to 82% for `Qwen3.5-4B`. The largest improvements occur for models where English-only transfer produces strong target-language calibration shifts, especially `Qwen3.5-4B`, `Qwen3-0.6B`, and `gpt-oss-low`.

These results show that pooled multilingual training substantially improves calibration across evaluated models. The improvement is especially important because English-only probes often remain discriminatively useful under transfer, but their predicted scores can become poorly calibrated for specific target languages. Pooled multilingual training does not merely improve isolated examples; it reduces mean ECE for every evaluated model and produces a more stable score scale across languages. Together with the transfer AUROC results in Section 4.2.2, these findings show that cross-lingual success prediction must be evaluated both as a ranking problem and as a calibration problem.

4.4. Multilingual Routing Results

This section evaluates whether the pre-generation success probes can support cost-aware multilingual model routing. The analysis first considers direct probe routing, where each candidate model provides its own routing representation, and then introduces encoder-target decoupled routing as a more cost-efficient alternative.

4.4.1. Direct Probe Routing

The first routing setting uses the model-specific probes analysed in the previous sections. For each candidate model, the router obtains a predicted success score from that model's own pre-generation activations and then applies the cost-aware routing policy described in Section 3.6.2. This setting is referred to as direct probe routing because the model used to provide the routing representation is also the target model.

Before comparing accuracy–cost frontiers, direct probe routing is evaluated across model depth in order to choose the probe layers used in the main comparison. For each relative layer depth, the routing parameters are selected on the validation split and the resulting policy is evaluated using the corresponding probe scores. Figure 4.7 reports the maximum validation success achieved by the pooled multilingual and English-only transfer routers at each depth.

The pooled multilingual router remains relatively stable across depth and reaches its strongest operating region at the final layer. In contrast, the English-only transfer router

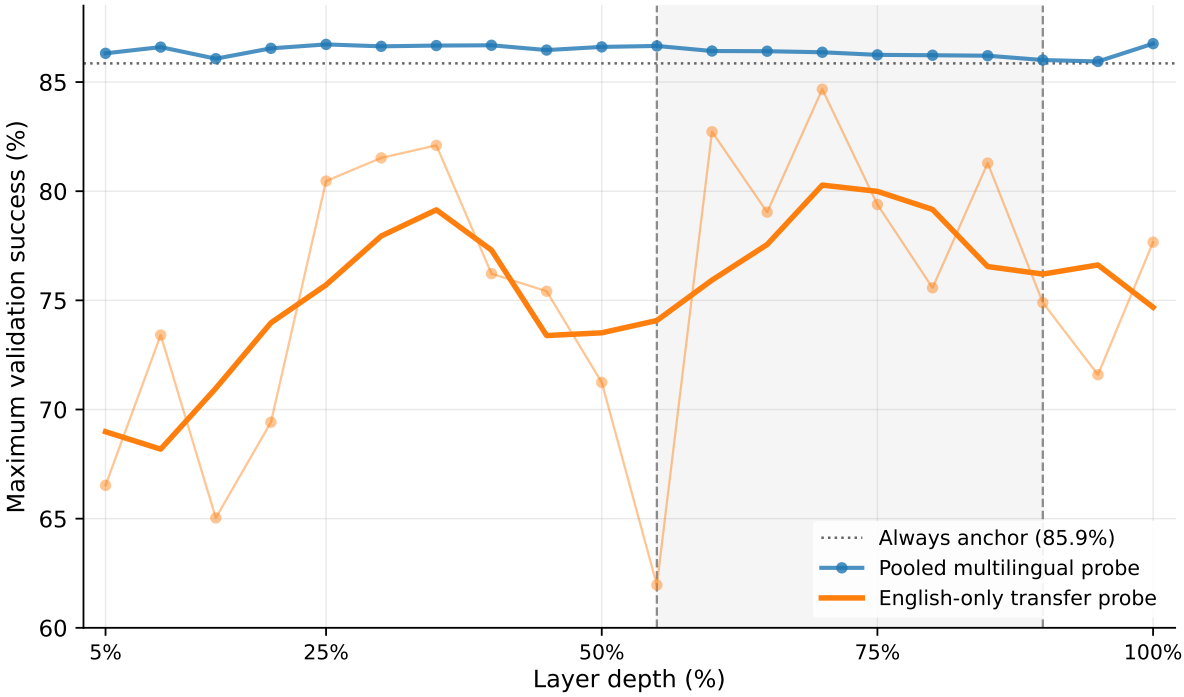


Figure 4.7: Layer-wise routing success for direct probe routing. Each point reports the maximum validation success obtained by applying the validation-selected routing policy at a given relative layer depth. The always-anchor baseline corresponds to always selecting `gpt-oss-high`.

is more sensitive to layer choice. Its validation success is lower in early layers, increases in the middle-to-late part of the network, and reaches its strongest value around 70% of model depth. The main direct-routing comparison therefore uses the final layer for the pooled multilingual probe and the 70%-depth layer for the English-only transfer probe.

Figure 4.8 shows the resulting test-set accuracy–cost trade-off under the full cost model. In this setting, direct routing requires the prompt to be processed by every candidate model before the routing decision is made, since each candidate model must provide its own activation-based success estimate. As a result, routing to cheaper models can reduce output cost only if the saving compensates for the candidate-model prefills required to compute the routing scores.

Table 4.7 reports two representative operating regimes. The highest-success point shows the maximum expected success achieved by each direct router. The anchor-matching point reports the operating point that reaches at least the expected success of always using `gpt-oss-high` while maximising cost saving.

The pooled multilingual direct router reaches 86.81% expected success at its highest-

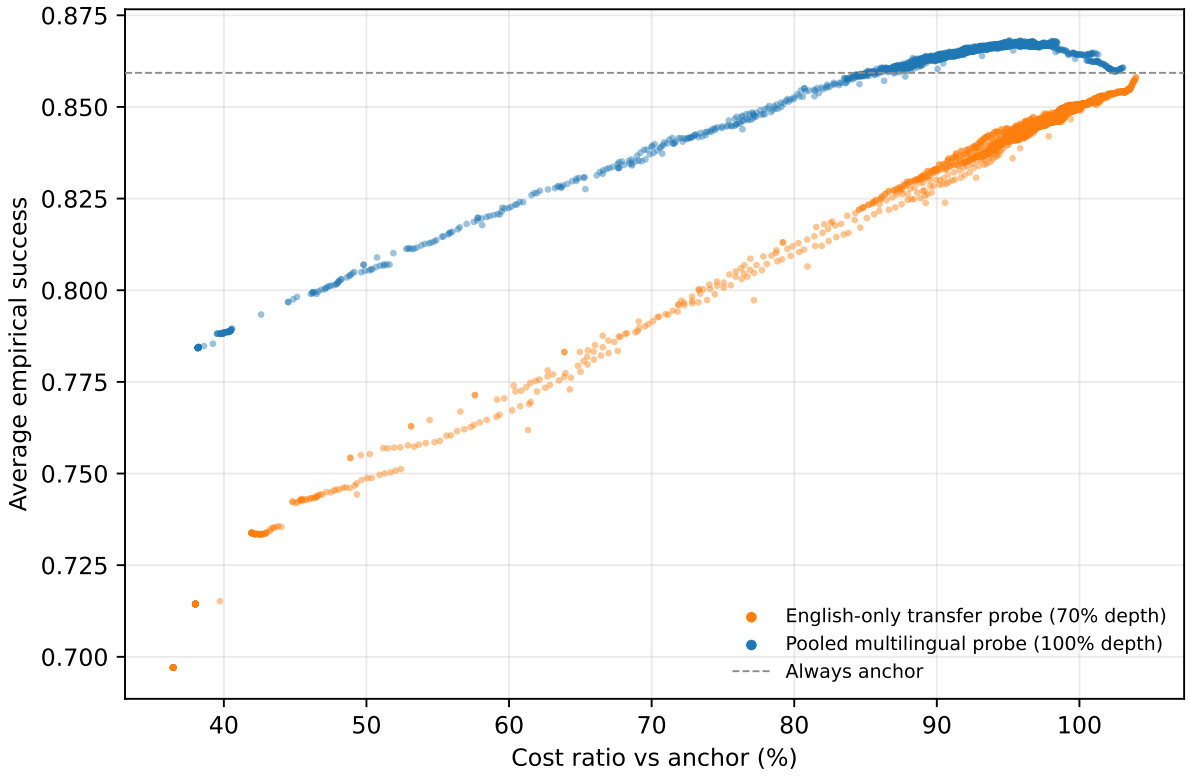


Figure 4.8: Accuracy–cost trade-off for direct probe routing on the test split. Each point corresponds to a routing policy obtained by varying the routing parameters. Cost is reported as a percentage of the expected cost of always using `gpt-oss-high`. The English-only transfer frontier uses the 70%-depth layer, while the pooled multilingual frontier uses the final layer.

success operating point, improving over the anchor by 0.88 percentage points while reducing evaluated cost by 4.94%. At the anchor-matching operating point, it reaches 85.98% expected success and saves 14.86% relative to always using `gpt-oss-high`.

The English-only transfer router does not reach an anchor-matching operating point. Its highest-success point reaches 85.81% expected success, which is 0.11 percentage points below the anchor, and is 3.97% more expensive than always selecting `gpt-oss-high`. Under the evaluated cost model, this operating point is therefore both slightly less accurate and more expensive than the anchor baseline.

The model-selection behaviour further describes the difference between the two direct routers. The pooled multilingual router uses a mixture of target models. At the anchor-matching point, it assigns 43.34% of examples to `gpt-oss-high`, with substantial fractions also routed to `Qwen3.5-9B`, `Qwen3.5-4B`, and `gpt-oss-low`. At its highest-success point, the same router assigns 65.18% of examples to `gpt-oss-high` and 25.82% to `Qwen3.5-9B`.

Setting	Success	Cost vs anchor	Delta Success vs anchor
Anchor baseline			
`gpt-oss-high`	85.93%	–	–
Highest-success operating point			
English-only transfer	85.81%	+3.97%	−0.11 pp
Pooled multilingual	86.81%	−4.94%	+0.88 pp
Anchor-matching operating point			
English-only transfer	Not achieved	Not achieved	Not achieved
Pooled multilingual	85.98%	−14.86%	+0.05 pp

Table 4.7: Representative operating points for direct probe routing on the test split. Success is the expected empirical success of the selected models, using the five-rollout success scores. Cost is measured relative to always selecting `gpt-oss-high`.

By contrast, the English-only transfer router routes 99.27% of examples to `gpt-oss-high` at its highest-success point. The full routing-mix tables are reported in Appendix B.4.

Overall, the direct-routing results show a clear separation between the pooled multilingual and English-only transfer settings under the full cost model. The pooled multilingual router reaches an anchor-matching operating point with positive savings and also improves over the anchor at its highest-success point. The English-only transfer router does not reach anchor success and increase cost due to the prefills at its best-success point.

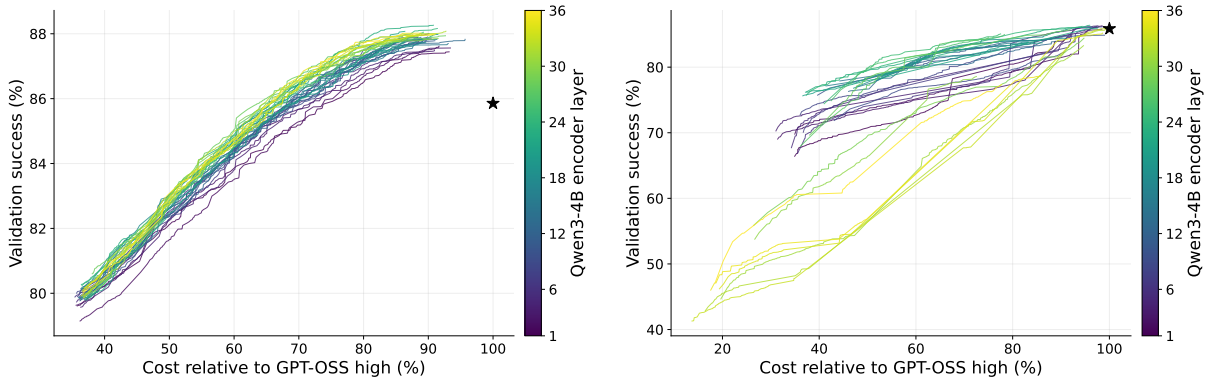
4.4.2. Encoder-Target Decoupled Routing

Encoder-target decoupled routing is evaluated as a more cost-efficient variant of probe-based routing. In this setting, the model used to extract the routing representation is decoupled from the target model selected for generation. The router first extracts the probe representation from a fixed encoder model and produces predicted success scores for all eight target models. The selected target model is then used for generation.

The main ETD results use `Qwen3-4B` as the encoder. This encoder is selected because it obtains the highest maximum validation success among the evaluated encoder choices, reaching 88.26%. The corresponding encoder layer is layer 23, which is approximately 64% of the model depth. The full comparison across encoder choices is reported in Appendix B.4.

To verify the layer choice for the ETD setting, the `Qwen3-4B` encoder is evaluated across all layers on the validation split. Figure 4.9 reports the resulting accuracy–cost frontiers

for pooled multilingual and English-only transfer ETD. In the pooled multilingual setting, the frontiers are relatively close across layers, although the strongest validation operating point is obtained at layer 23. In the English-only transfer setting, the frontiers vary more visibly with encoder depth: early layers remain below the strongest region, the best trade-offs are concentrated in the middle-to-late part of the model, and performance declines again in the final layers. This shape is consistent with the layer-wise transfer analysis in Section 4.2.2, where cross-lingual probe transfer is strongest before the final layers. Layer 23 is therefore used for both ETD routing policies in the main comparison.



(a) Pooled multilingual ETD (b) English-only transfer ETD

Figure 4.9: Validation accuracy–cost frontiers for ETD routing across `Qwen3-4B` encoder layers. Each curve corresponds to one encoder layer. Cost is reported as a percentage of the evaluated cost of always selecting `gpt-oss-high`. The pooled multilingual setting shows smaller variation across layers, while the English-only transfer setting is more sensitive to encoder depth.

Figure 4.10 and Table 4.8 report the main ETD routing results on the test split. The figure shows the full accuracy–cost frontiers obtained by varying the routing parameters, while the table reports two representative operating points for each router: the highest-success point and the anchor-matching point. Both routers use `Qwen3-4B` as the encoder, layer 23 as the probe layer, and the same set of target models.

The ETD results preserve the same ordering between pooled multilingual and English-only transfer routing observed in the direct-routing setting. The English-only ETD router now reaches operating points above the anchor baseline, but its high-success region remains close to the always-anchor cost level. Its highest-success point obtains 86.07% success with a 3.59% cost reduction, while its anchor-matching point obtains 85.95% success with a 4.87% cost reduction. Moving toward lower-cost operating points leads to a visible decrease in success along the frontier.

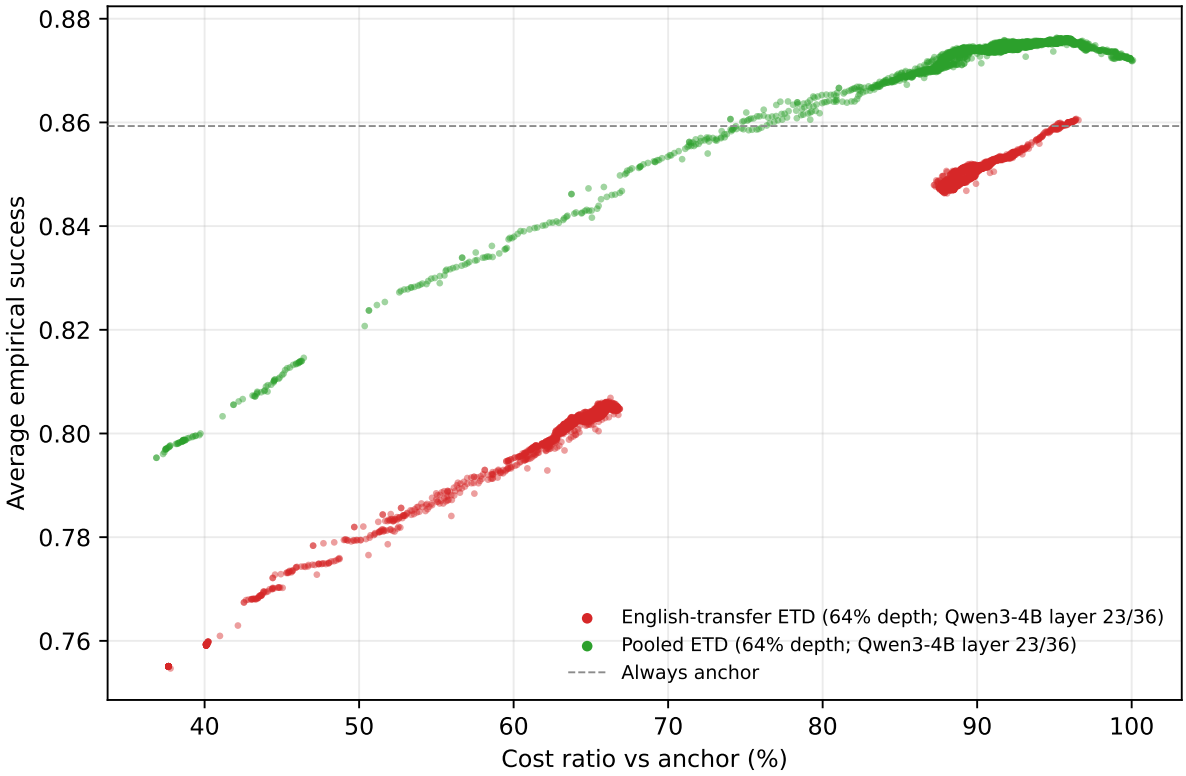


Figure 4.10: Accuracy–cost trade-off for encoder-target decoupled routing on the test split. Both routers use `Qwen3-4B` as the encoder and layer 23 as the probe layer. Cost is reported as a percentage of the evaluated cost of always selecting `gpt-oss-high`.

The pooled multilingual ETD frontier extends further toward both higher success and lower cost. It reaches 87.63% success at its highest-success point, improving over the anchor by 1.70 percentage points while reducing evaluated cost by 4.83%. At the anchor-matching point, it reaches 86.06% success and reduces evaluated cost by 25.97%. In the high-success region of Figure 4.10, the pooled multilingual frontier therefore includes both higher-success points and lower-cost anchor-level points than the English-only ETD frontier.

The model-selection mix also differs between the two routers. At the pooled ETD anchor-matching point, 28.57% of examples are routed to `gpt-oss-high`, 26.64% to `Qwen3.5-9B`, 19.36% to `Qwen3.5-4B`, and 18.45% to `gpt-oss-low`. At the pooled ETD highest-success point, the router assigns 75.89% of examples to `gpt-oss-high` and 12.75% to `Qwen3.5-9B`. The English-only ETD router is more concentrated on the anchor model, assigning 82.48% of examples to `gpt-oss-high` at the anchor-matching point and 87.25% at the highest-success point. The complete ETD routing-mix tables are reported in Appendix B.4.

Figure 4.11 compares the pooled multilingual ETD router with the pooled multilingual

Setting	Success	Cost vs anchor	Delta Success vs anchor
Anchor baseline			
`gpt-oss-high`	85.93%	–	–
Highest-success operating point			
English-only ETD	86.07%	−3.59%	+0.14 pp
Pooled multilingual ETD	87.63%	−4.83%	+1.70 pp
Anchor-matching operating point			
English-only ETD	85.95%	−4.87%	+0.02 pp
Pooled multilingual ETD	86.06%	−25.97%	+0.13 pp

Table 4.8: Representative operating points for encoder-target decoupled routing on the test split. Both ETD routers use `Qwen3-4B` as the encoder and layer 23 as the probe layer. Success is the expected empirical success of the selected models, using the five-rollout success scores. Cost change is measured relative to the evaluated cost of always selecting `gpt-oss-high`; negative values indicate savings, while positive values indicate higher cost.

direct router. This comparison keeps the pooled multilingual training setting fixed and changes only how the routing scores are obtained. The direct router computes probe scores from all candidate target-model prefills, whereas the ETD router computes probe scores from a shared `Qwen3-4B` encoder before target-model selection.

The comparison highlights the effect of candidate-model prefill cost. At the highest-success operating point, the direct pooled router reaches 86.81% success with a 4.94% cost reduction, while the pooled ETD router reaches 87.63% success with a similar cost reduction of 4.83%. In this region, ETD therefore reaches a higher success point at nearly the same evaluated cost.

The difference is larger at the anchor-matching operating point. The direct pooled router reaches 85.98% success with a 14.86% cost reduction, whereas the pooled ETD router reaches 86.06% success with a 25.97% cost reduction. Since both routers use pooled multilingual probes, this comparison isolates the effect of replacing candidate-model prefills with a shared encoder representation before target-model selection.

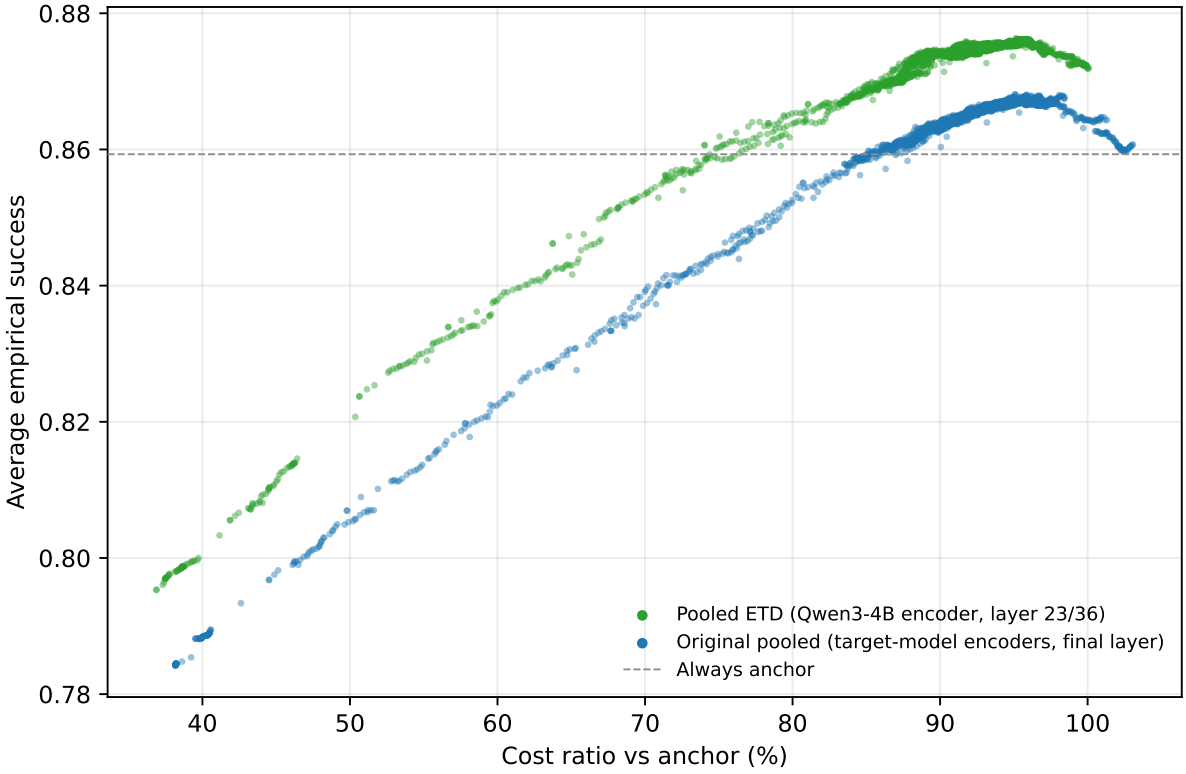
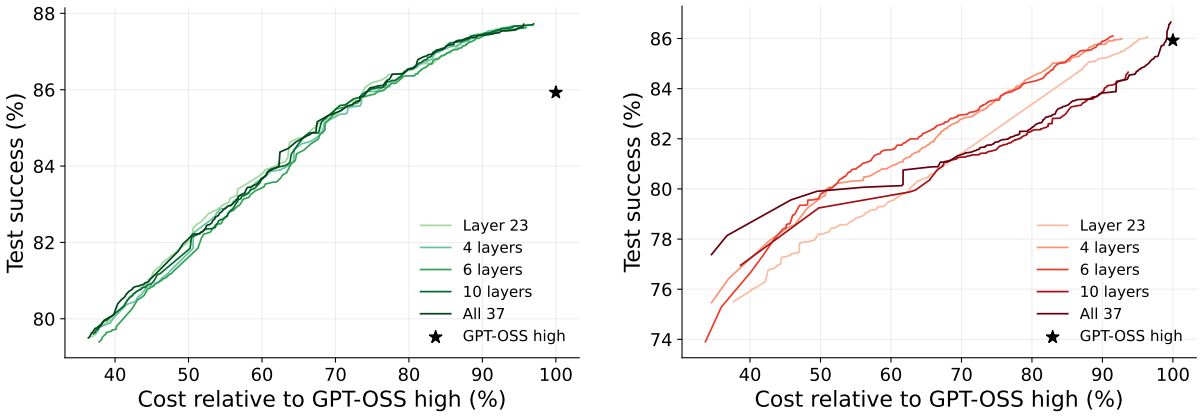


Figure 4.11: Accuracy–cost trade-off for pooled multilingual routing with direct and encoder-target decoupled probe extraction. The direct router uses target-model encoders at the final layer, while the ETD router uses `Qwen3-4B` layer 23 as a shared encoder.

4.4.3. Layer Ensembles

The final routing comparison evaluates whether ETD routing benefits from combining probe predictions across multiple encoder layers. For each ensemble, one ridge probe is trained at each selected `Qwen3-4B` encoder layer. The predicted success scores are then averaged across layers and the averaged score is used inside the same routing policy. The ensemble therefore combines probe outputs rather than concatenating hidden representations. The reported ensembles use predefined evenly spaced layer sets: the single-layer baseline uses layer 23; the 4-layer ensemble uses layers 18, 24, 30, 36; the 6-layer ensemble uses layers 21, 24, 27, 30, 33, 36; the 10-layer ensemble uses layers 9, 12, 15, 18, 21, 24, 27, 30, 33, 36; and the all-layer ensemble uses all layers from 0 to 36.

Figure 4.12 reports the layer-ensemble results for pooled multilingual and English-only ETD routing. The figure shows the full test-set accuracy–cost frontiers, while Tables 4.9 and 4.10 report the highest-success and anchor-matching operating points for each layer set.



(a) Pooled multilingual ETD (b) English-only transfer ETD

Figure 4.12: Test-set accuracy–cost frontiers for ETD routing with layer-ensemble probes. Each ensemble averages predicted success scores across the selected `Qwen3-4B` encoder layers before applying the routing policy. Cost is reported as a percentage of the evaluated cost of always selecting `gpt-oss-high`.

Layer set	Max success	Cost at max	Matched success	Cost at match
Layer 23	87.63%	−4.83%	86.06%	<u>−25.97%</u>
4 layers	87.69%	−3.18%	85.95%	−25.70%
6 layers	87.63%	−4.07%	85.94%	−24.34%
10 layers	<u>87.73%</u>	−3.04%	85.94%	−25.87%
All 37 layers	87.71%	−4.37%	85.99%	−25.52%

Table 4.9: Representative operating points for pooled multilingual ETD routing with layer-ensemble probes on the test split. Success is the expected empirical success of the selected models, using the five-rollout success scores. Cost is measured relative to the evaluated cost of always selecting `gpt-oss-high`; negative values indicate savings. Underlined values mark the highest maximum success and the largest anchor-matching saving.

For pooled multilingual ETD, layer ensembling changes the frontier only marginally. The best maximum success is obtained by the 10-layer ensemble, which reaches 87.73%, compared with 87.63% for the single layer-23 router. However, this gain is small, and no pooled ensemble improves the anchor-matching cost reduction obtained by layer 23. The single-layer router saves 25.97% at the anchor-matching point, while the closest ensemble is the 10-layer setting with a 25.87% cost reduction. This is consistent with the layer-wise ETD results in Figure 4.9, where pooled multilingual routing is relatively stable across encoder depth.

Layer set	Max success	Cost at max	Matched success	Cost at match
Layer 23	86.07%	−3.59%	85.95%	−4.87%
4 layers	85.99%	−7.24%	85.95%	−8.40%
6 layers	86.10%	−8.51%	85.95%	−9.49%
10 layers	84.65%	−6.35%	Not achieved	Not achieved
All 37 layers	86.65%	−0.31%	85.94%	−0.95%

Table 4.10: Representative operating points for English-only transfer ETD routing with layer-ensemble probes on the test split. Success is the expected empirical success of the selected models, using the five-rollout success scores. Cost is measured relative to the evaluated cost of always selecting `gpt-oss-high`; negative values indicate savings. Underlined values mark the highest maximum success and the largest anchor-matching saving.

The contrast is clearer in the English-only transfer setting. The 4-layer and 6-layer ensembles improve the trade-off relative to the single layer-23 router, shifting the frontier toward lower evaluated cost while preserving similar high-success operating points. The 6-layer ensemble provides the strongest practical improvement: it reaches 86.10% maximum success with an 8.51% cost reduction, compared with 86.07% success and a 3.59% cost reduction for layer 23. At the anchor-matching point, it also increases the cost reduction from 4.87% to 9.49%. By contrast, adding layers outside the strongest middle-to-late region weakens the trade-off. The 10-layer ensemble does not reach an anchor-matching point, while the all-layer ensemble reaches 86.65% maximum success only near the anchor cost level, with a cost reduction of 0.31%.

4.5. Cross-dataset Evaluation on BFCL

This section evaluates whether the previous success-prediction findings also transfer to BFCL, a structurally different function-calling benchmark. The goal is to test whether pre-generation success prediction also applies outside mathematical reasoning when task-relevant supervision is available.

4.5.1. Empirical Success on BFCL

Table 4.11 reports the empirical BFCL success rates of the Qwen-family models. These values provide the task-level performance context for the subsequent probing analysis. The reported metric is the mean empirical success over five sampled generations per prompt, $k/5$, averaged over the full aligned BFCL evaluation set.

Model	BFCL success (%)
`Qwen3-0.6B`	65.71
`Qwen3-1.7B`	78.32
`Qwen3-4B`	**84.76**
`Qwen3-8B`	82.91
`Qwen3.5-4B`	70.70
`Qwen3.5-9B`	69.25

Table 4.11: BFCL success is mean empirical success over five sampled generations per prompt ($k/5$), reported on the full aligned BFCL evaluation set of 3251 examples per model across 10 BFCL subsets.

The results show that BFCL performance varies substantially across the Qwen-family models. `Qwen3-4B` obtains the highest mean empirical success rate at 84.76%, followed by `Qwen3-8B` at 82.91% and `Qwen3-1.7B` at 78.32%. The smaller `Qwen3-0.6B` model performs noticeably worse, while the two Qwen3.5 models obtain lower overall BFCL success than the strongest Qwen3 models in this evaluation.

4.5.2. Success Prediction Across Datasets

The experiment evaluates whether the success-prediction findings from the MATH setting extend to BFCL. This analysis is not intended to claim that a single probe should transfer across arbitrary tasks. Instead, BFCL is used as a structurally different benchmark to test whether pre-generation activations also contain success-predictive information outside mathematical reasoning.

The experiment compares three probe-training configurations. The first trains a probe on MATH and evaluates it directly on BFCL. This setting tests direct cross-dataset transfer. The second trains a probe jointly on MATH and BFCL and evaluates it on BFCL. The third trains and evaluates a probe on BFCL, providing a task-specific reference. All three configurations use the same layer-wise ridge-regression procedure and AUROC evaluation used in the main probing experiments.

Figure 4.13 shows that success is linearly decodable from pre-generation activations in the BFCL setting when BFCL supervision is available. The BFCL-only probe remains consistently above chance across the model depth, with AUROC values concentrated around the upper part of the plotted range. This indicates that the model's pre-generation representation contains information about whether the generated function call is likely to be correct.

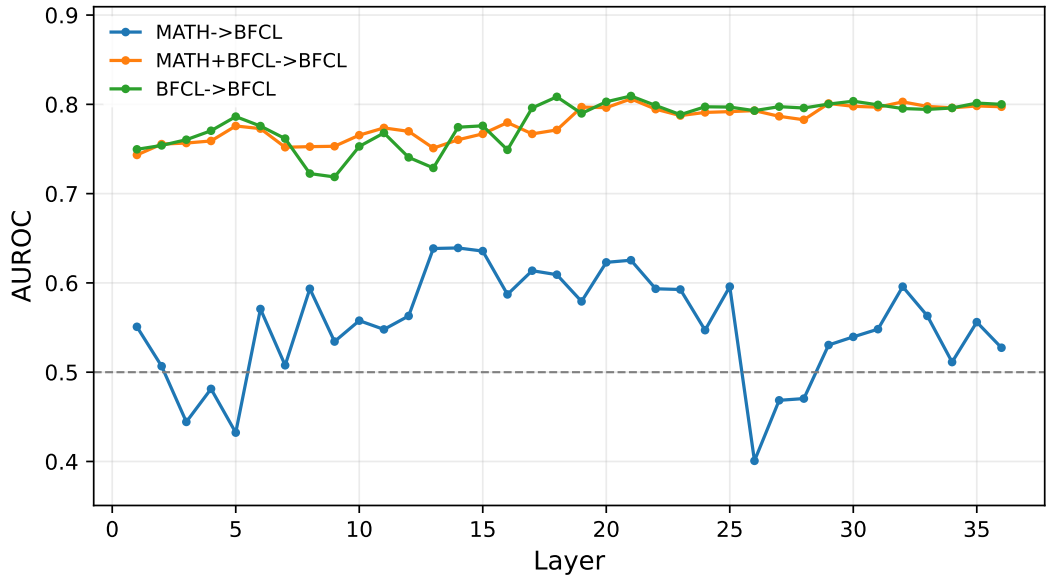


Figure 4.13: Layer-wise BFCL AUROC for `Qwen3-8B` under three training configurations. The MATH-only probe is trained on MATH and evaluated on BFCL. The joint probe is trained on MATH and BFCL and evaluated on BFCL. The BFCL-only probe is trained and evaluated on BFCL. The dashed line indicates chance-level AUROC.

The joint MATH+BFCL probe closely follows the BFCL-only probe across layers. This shows that adding MATH examples during training does not prevent the probe from learning BFCL-relevant success information. In this setting, success prediction therefore extends cleanly to BFCL when the training data include BFCL supervision.

Direct transfer from MATH to BFCL is weak and unreliable compared with BFCL-supervised prediction. The result is consistent with the difference between the two datasets: MATH success depends on producing a correct mathematical answer, while BFCL success depends on producing a correct structured function call.

The same qualitative pattern is observed across the Qwen-family models, with the full layer-wise trajectories reported in Appendix B.5. Pre-generation success signals are not limited to the MATH benchmark, since they can also be decoded in a function-calling setting. However, the direct MATH to BFCL probe does not provide a robust task-general predictor, suggesting that reliable success prediction across datasets requires task-relevant supervision.

5 | Discussion

This chapter interprets the empirical results, focusing on the implications of cross-lingual transfer, layer-wise structure, calibration, and routing behaviour in multilingual pre-generation success prediction.

5.1. Overview of Findings

The results show that pre-generation hidden activations contain information about a model's future success across the evaluated models and languages. In the same-language setting, the probes achieve consistently informative AUROC values, showing that success is linearly decodable from the prompt-conditioned representation before any answer tokens are generated. This supports the central premise of the thesis: model success is not only visible after generation, but is already partly reflected in the internal state formed after the model has read the prompt.

The cross-lingual results show that this success signal is not purely language-specific. Probes trained in one language remain discriminative when evaluated on other languages, with peak transfer AUROC values substantially above random across all models. The gap between same-language and cross-lingual AUROC is relatively small at the transfer-optimal layer, indicating that a large part of the success-prediction signal transfers across languages. The results therefore indicate that pre-generation success probes generalise reasonably well across languages, while still performing best when the training and evaluation language match.

The layer-wise analysis adds an important qualification to this finding. Same-language prediction reaches strong performance in middle-to-late layers and remains close to its peak through the final layers. Cross-lingual transfer, by contrast, peaks in a narrower middle-to-late depth range and then declines more sharply toward the final layers. This pattern is consistent with the view that later representations remain useful for within-language prediction but become less aligned across languages as the model approaches generation. The evidence is probe-based rather than causal, so it does not prove that the

model uses these representations in a particular way. However, it suggests that transferable success information is most accessible before the final hidden states.

The calibration results show that discriminative transfer is not sufficient for reliable multilingual success prediction. English-trained probes can transfer as ranking functions, but their predicted scores are not consistently calibrated across target languages. Pooled multilingual training substantially reduces this calibration error by exposing the probe to activation distributions from all evaluated languages, even though the probe is not given language identity as an explicit input. This suggests that multilingual training is one effective way to improve calibration across languages.

The routing results show that these differences matter for downstream model selection. English-only transfer probes are too weak to provide a reliable standalone multilingual routing signal. They can rank examples to some extent, but their score scale is not robust enough to consistently identify when cheaper models should replace the strongest model. Pooled multilingual probes are more effective because they provide a more reusable and better calibrated basis for comparing candidate models across languages. This is clearest in the encoder-target decoupled routing setting, where a shared `Qwen3-4B` encoder is used to compute routing scores before selecting the target model for generation. With this approach, pooled multilingual routing reaches 87.63% success at its highest-success operating point while still saving 4.83% of cost, and reaches 86.06% success at the anchor-matching point with a cost saving of 25.97%. These results show that multilingual success probes can support cost-aware model selection, provided that both score reliability and score-extraction cost are taken into account.

The BFCL evaluation tests whether the success-prediction findings are also observed on a different benchmark. The results show that success is linearly decodable in the function-calling setting when BFCL supervision is available. However, direct transfer from MATH to BFCL is weak and unstable across layers. This suggests that the success signal is not fully task-general, and that applying success prediction to a different dataset requires task-relevant supervision.

Overall, the experiments show that multilingual pre-generation success prediction has two separable requirements. The probe must recover a success-related signal that transfers across languages, and its score scale must remain reliable enough to support probability-like interpretation and downstream decisions. The following sections discuss these findings in relation to cross-lingual representation transfer, layer-wise structure, model competence, calibration and multilingual routing.

5.2. Cross-lingual Transfer of Success Signals

The cross-lingual transfer results address the central question of whether success probes learn only language-specific patterns or whether they recover a signal that generalises across translated versions of the same task. The results support the second interpretation. Probes trained on one language remain discriminative when evaluated on other languages, with transfer AUROC well above random across all evaluated models. This shows that the success-related information available in pre-generation activations is not confined to the language used during probe training.

This finding is important because the probe is not given access to the original problem text, the target language identity, or the generated answer. It receives only the hidden activation vector produced after the model processes the prompt. Cross-lingual transfer therefore suggests that the activation space contains a success-related structure that is at least partly shared across languages. A probe trained on one language can still identify patterns associated with likely success or failure in another language, even though the surface form, script, tokenisation, and prompt length may differ.

At the same time, transfer is not identical to same-language prediction. Across evaluated models, cross-lingual AUROC remains consistently below the same-language reference at the transfer-optimal layer. The gap is relatively small, ranging from 0.020 to 0.060 AUROC, which indicates that transfer is reasonably strong. However, the gap is also consistent across evaluated models, showing that some predictive information is easier to recover when the probe is trained and evaluated in the same language.

This result fits the broader tension in multilingual representation learning between shared semantic structure and language-specific variation. Since all language versions correspond to the same underlying mathematical problems, a fully language-invariant representation would make cross-lingual probing nearly equivalent to same-language probing. The observed transfer gap suggests that this is not the case. The model appears to preserve enough shared task-level or difficulty-related information for transfer to work, while also retaining language-dependent structure that affects the exact predictive direction learned by the probe.

The results therefore support a practical view of cross-lingual transfer. A success probe trained in one language can provide useful discriminative information in other languages, but the best performance is still obtained when the training and evaluation language match. This matters for multilingual deployment because it suggests that labelled rollout data from one language may remain useful when moving to another language. However,

cross-lingual reuse still comes with a measurable performance cost, and that cost should be evaluated rather than assumed away.

This interpretation is especially relevant for languages where the base model is weaker. The base model results showed large differences in empirical success rates across languages, with Swahili and Telugu among the lowest-performing settings. The fact that cross-lingual probes still transfer in this multilingual setting suggests that the probe is not simply memorising English-specific or high-resource-language patterns. Instead, it recovers information that remains useful across a diverse set of languages. However, as discussed later in this chapter, transferring a ranking signal is not the same as transferring a reliable probability scale.

5.3. Layer-wise Structure of Transfer

The layer-wise results show that success prediction is not distributed uniformly across model depth. In the same-language setting, probe performance generally increases through the early layers and remains strong across the middle-to-late layers. In the cross-lingual setting, transfer performance is more depth-sensitive. It improves after the early layers, reaches its strongest values in the middle-to-late part of the model, and often declines toward the final layers.

This pattern suggests that the most transferable success-related information is not necessarily found in the final hidden state. The final layers are closest to generation and may encode information that is more strongly shaped by the prompt language, the model's response policy, or language-specific output preparation. By contrast, intermediate and late-intermediate layers may preserve a representation in which task difficulty and model competence are still more accessible across translated versions of the same problem.

For same-language prediction, later layers remain useful. Once the probe reaches strong AUROC, performance forms a broad plateau rather than collapsing near the output side of the model. This indicates that hidden states in the later layers continue to contain information that separates examples the model tends to solve from examples it tends to fail, at least when the probe is trained and evaluated in the same language. In this setting, language-specific structure does not prevent prediction because the training and evaluation distributions are matched.

Cross-lingual transfer behaves differently. Transfer performance peaks in a narrower middle-to-late depth range, between 59% and 71% of network depth across the evaluated models, and then decreases toward the final layer. This decline is larger than the

corresponding same-language decline. The result suggests that the most transferable success signal is not located at the final hidden state, even though final-layer representations remain useful for same-language prediction.

A plausible interpretation is that intermediate and late-intermediate layers preserve a better balance between task-level information and language-specific information. In mathematical reasoning prompts, task-level information includes aspects such as the structure of the problem, the operations required, and the model's apparent likelihood of solving it. These aspects may be shared across translated versions of the same problem. Later layers, by contrast, are closer to the point of answer generation. At this stage, representations may become more specialised for producing an answer in the prompt language, including language-specific lexical, syntactic, or formatting choices. This would explain why same-language prediction remains strong while cross-lingual transfer weakens.

The result also has methodological implications. If success probes are evaluated only at the final layer, the transferability of the success signal may be underestimated. Final-layer representations are convenient because they are often treated as the model's final prompt representation before generation, but they are not necessarily the best representation for cross-lingual prediction. A layer-wise analysis is therefore necessary to identify where transferable information is most accessible. In this thesis, the optimal transfer layers occur before the final layers for every evaluated model.

At the same time, the evidence should be interpreted carefully. The probes show where success information is linearly decodable, but they do not establish that the model itself uses the same information causally during generation. The final-layer decline in transfer is consistent with later layers becoming more language-specific, but it does not directly prove the mechanism responsible for the decline. Other factors, such as probe alignment or the distribution of activation directions across languages, may also contribute. The routing-by-depth analysis provides an additional downstream view of the same layer-wise structure. English-only transfer routing is strongest around the same middle-to-late depth region where cross-lingual AUROC is strongest, whereas pooled multilingual routing is much less sensitive to layer choice. The main conclusion is therefore representational and operational. Transferable success information is most linearly accessible before the final layers, and this depth dependence can also affect downstream routing when the router relies on English-only transfer scores.

5.4. Success Prediction, Difficulty, and Model Competence

The results show that success prediction is not the same as measuring raw model performance. Base success rates describe how often a model solves problems in each language, while probe AUROC describes how well the hidden representations separate examples the model tends to solve from examples it tends to fail. These two quantities are related, but they are not equivalent. A language or model can have low raw success while still producing highly separable success and failure representations, and a strong model can achieve high success while leaving fewer or more subtle failures for the probe to distinguish.

This distinction is visible across the evaluated models. The smallest model, `Qwen3-0.6B`, has the lowest average base success rate, but it achieves the highest peak same-language AUROC and the highest peak transfer AUROC. Conversely, the strongest evaluated settings by raw success do not necessarily produce the highest probe AUROC. This does not mean that the smaller model is better at mathematical reasoning. Rather, it suggests that its successes and failures are more easily separated in activation space. For stronger models, the remaining failures may occur on examples that are closer to the model's decision boundary, making them harder to distinguish linearly from successful cases.

A similar pattern appears at the language level. Swahili and Telugu have among the lowest base success rates, yet they show some of the highest same-language AUROC values. This indicates that low performance does not necessarily make success prediction harder. In these languages, the model may fail more often, but those failures may be more systematic or more clearly reflected in the pre-generation hidden state. By contrast, languages with higher base success rates can produce a less polarised prediction problem if successes and failures occupy more overlapping regions of representation space.

These observations suggest that empirical success reflects several interacting factors. Part of the variation comes from the intrinsic difficulty of the mathematical problem: some problems require more steps, more abstract reasoning, or more precise calculations. Another part comes from the language in which the problem is expressed. Translation choices, script, tokenisation, and prompt length can all affect how the model represents the same underlying problem. A third part comes from model-language competence: the same model may be more capable in some languages than in others because of training data coverage, tokenizer suitability, or general multilingual ability.

The probe is trained on the outcome of all these factors together. It does not observe an isolated measure of mathematical difficulty, nor does it receive an explicit variable

describing language competence. It only receives the pre-generation activation produced by a particular model for a particular language version of a problem. A high AUROC therefore means that the activation contains information predictive of the model's empirical success in that setting. It does not by itself identify whether the relevant signal corresponds to problem-intrinsic difficulty, language-induced difficulty, model uncertainty, or some combination of these.

This matters for interpreting cross-lingual transfer. If the probe were learning only problem-intrinsic difficulty, then translated versions of the same problem would ideally produce very similar success predictions across languages. However, the base success rates show that the same mathematical problems can have different empirical outcomes depending on language and model. Cross-lingual success prediction therefore requires more than recognising the underlying mathematical problem. It also requires the success-related representation to remain stable despite language-specific shifts in how the model processes the prompt.

The results suggest that this stability is partial. Transfer AUROC remains strong, which indicates that the probe captures some structure that generalises across languages. At the same time, the same-language advantage and the final-layer transfer decline show that the success signal is not purely problem-intrinsic. It is shaped by the interaction between the problem, the language, and the model. This supports the view that pre-generation success prediction should be interpreted as model- and language-conditioned difficulty estimation, rather than as a universal measure of task difficulty.

5.5. Discrimination and Calibration in Multilingual Probing

The calibration results show that successful cross-lingual probing cannot be evaluated by AUROC alone. AUROC measures whether a probe assigns higher scores to examples the model tends to solve than to examples it tends to fail. This is a ranking property. Calibration asks a different question: whether the numerical score produced by the probe corresponds to the empirical probability of success. A probe can therefore be useful for ranking prompts by expected difficulty while still being unreliable as a probability estimator.

This distinction is central to the multilingual setting. The cross-lingual AUROC results show that probes trained in one language retain discriminative information when evaluated on other languages. However, the English-trained reliability diagrams show that the

same score scale does not transfer reliably across all target languages. English provides a relatively well-calibrated in-language reference, but several transferred languages show substantially higher calibration error. This means that the probe can still recognise relative differences between easier and harder examples, while assigning predicted scores that do not match the target language's observed success rates.

The failure of English-only calibration is consistent with the base model performance differences across languages. English has the highest average success rate across models, while languages such as Swahili and Telugu have much lower success rates. A probe trained only on English learns a score scale shaped by the English activation distribution and English success frequencies. When this score scale is applied to another language, the ranking signal may remain useful, but the absolute values may become too optimistic or too conservative. In this sense, calibration is more sensitive than discrimination to shifts in the target distribution.

Pooled multilingual training improves this situation by exposing the probe to multilingual variation during training. Since the pooled probe is trained without explicit language identity, the improvement cannot be attributed to a simple language-specific correction term. Instead, the probe learns a score scale that is more compatible with the range of success frequencies observed across languages. The reduction in ECE across all evaluated models suggests that multilingual training is one effective way to improve calibration in this setting.

The size of the calibration improvement varies across evaluated models. The largest gains occur in settings where English-only transfer is most miscalibrated, especially `Qwen3.5-4B`, `Qwen3-0.6B`, and `gpt-oss-low`. This indicates that the benefit of pooled multilingual training depends less on model size alone than on how poorly the English-trained score scale matches the multilingual target distributions. Models with lower English-only ECE still benefit from pooled training, but the relative improvement is smaller because there is less calibration error to correct. This suggests that stronger cross-lingual calibration is not automatic, even when the discriminative transfer signal is present.

These findings clarify what kind of transfer is needed for practical use. A cross-lingual success probe may be sufficient if the goal is only to rank prompts by relative likelihood of success. However, if the probe score is used as a probability, or if a fixed threshold is applied across languages, calibration becomes essential. The results show that English-only probes should not be assumed to provide reliable multilingual probability estimates, even when their AUROC is strong. Multilingual training provides a more reliable alternative, but its reliability is demonstrated here only within the evaluated set of languages and

models.

5.6. Implications for Multilingual Routing

The routing results indicate that the usefulness of pre-generation success probes depends not only on their ability to discriminate between likely successes and failures, but also on whether their scores can support stable decision-making under cost constraints. This distinction is important because routing is a thresholded and cost-sensitive use case. A probe that ranks examples reasonably well may still lead to weak routing behaviour if its score scale does not support reliable comparisons across languages or models.

The English-only transfer router shows the limitation of relying on English supervision alone. The preceding analysis established that English-trained probes can transfer discriminative information across languages, especially in the middle-to-late layers. However, the routing experiment indicates that discriminative transfer is not sufficient by itself. For routing, the probe scores must be reliable enough to compare candidate models and select the model with the most appropriate accuracy–cost trade-off for each prompt. English transfer does not provide this reliability in the evaluated setting. Its routing frontier remains close to the strongest-model baseline and identifies only a limited set of cases where alternative candidate models can be selected without reducing expected success.

This result clarifies the difference between transferable ranking and useful routing. English-only probes retain some cross-lingual ranking signal, but their score scale is too unstable to support strong threshold-based decisions across languages. This is consistent with the calibration results: a probe may assign higher scores to examples that are more likely to be solved while still producing numerical values that are not reliable enough for cost-sensitive model selection. In practical terms, English-only probes are therefore too weak to be used directly as a standalone multilingual routing signal.

The pooled multilingual router behaves differently. Because the probe is trained on examples from all evaluated languages, it learns from a broader multilingual activation distribution and produces scores that are better suited to threshold-based decisions. In the direct pooled routing setting, the router reaches 86.81% expected success at the highest-success operating point with a cost saving of 4.94%. At the anchor-matching operating point, it reaches 85.98% expected success with a cost saving of 14.86%. These results show that pooled multilingual training changes the routing trade-off rather than merely reproducing the strongest-model baseline. It can preserve anchor-level success while routing a meaningful fraction of examples to cheaper models.

The comparison between English-only and pooled multilingual routing connects the routing results directly to the calibration findings. Earlier results showed that pooled multilingual training produces a more reliable score scale than English transfer. The routing experiment shows why this matters operationally: cost-aware routing requires scores that can be used as decision values, not only as rankings. A better-calibrated multilingual score scale makes it possible to replace the strongest model selectively while preserving expected success. The benefit of pooled multilingual training is therefore visible not only in reliability diagrams and ECE values, but also in the downstream accuracy–cost trade-off.

At the same time, direct probe-based routing has an important efficiency limitation. If each candidate model must process the prompt before its probe score can be computed, then the router has already incurred the prompt-processing cost for the full model pool before selecting which model should generate the answer. Encoder-target decoupling addresses this limitation by separating the model used to obtain the routing representation from the model selected for generation. In this setting, a shared `Qwen3-4B` encoder is used to estimate the expected success of all candidate target models, and only the selected target model is then used for generation.

The ETD results strengthen the same conclusion. With pooled multilingual ETD, the router reaches 87.63% expected success at its highest-success operating point while achieving a cost saving of 4.83%. At the anchor-matching operating point, it reaches 86.06% expected success, effectively matching the anchor baseline, while achieving a cost saving of 25.97%. Compared with direct pooled routing, ETD therefore preserves the value of the multilingual routing signal while improving the cost-saving regime. The gain is especially clear at the anchor-matching point, where the cost saving increases from 14.86% to 25.97%.

English-only ETD improves over direct English-only routing, but it remains much weaker than pooled multilingual ETD. Its highest-success operating point reaches 86.07% expected success with a cost saving of 3.59%, and its anchor-matching point reaches 85.95% expected success with a cost saving of 4.87%. These values show that ETD can make score extraction more cost-effective, but it does not remove the underlying limitation of English-only supervision. The English-only probe still needs pooled multilingual support to become a robust routing signal across languages.

The model-selection distributions support the same interpretation. At the pooled ETD highest-success operating point, the router selects `gpt-oss-high` for 75.89% of examples and `Qwen3.5-9B` for 12.75%, reflecting a conservative policy that still improves over the

anchor. At the anchor-matching point, the router selects `gpt-oss-high` for only 28.57% of examples and distributes many remaining examples across intermediate models, especially `Qwen3.5-9B`, `Qwen3.5-4B`, and `gpt-oss-low`. By contrast, the English-only ETD router remains more concentrated on the anchor, selecting `gpt-oss-high` for 87.25% of examples at the highest-success point and 82.48% at the anchor-matching point. This suggests that the pooled router is not simply favouring lower cost; it identifies a larger subset of examples for which non-anchor models are expected to be sufficient.

Layer ensembling provides a useful additional observation. For pooled multilingual ETD, ensembling changes the frontier only marginally, which is consistent with the greater stability of pooled probes across depth. For English-only ETD, the effect is more informative. The 4-layer and 6-layer ensembles improve the accuracy–cost trade-off compared with the single layer-23 router, indicating that nearby layers in the middle-to-late region contain complementary transferable routing information. However, ensembling is not automatically beneficial. When the ensemble is expanded to include layers outside this effective region, as in the 10-layer and all-layer settings, the routing frontier becomes worse. This suggests that adding less transferable layers can dilute the useful signal rather than stabilise it. The result is consistent with the broader layer-wise analysis: English-only transfer is strongly layer-dependent, so focused ensembling around the transfer-effective region can improve routing, whereas overly broad ensembling weakens the trade-off. Overall, these improvements still do not close the gap to pooled multilingual ETD, suggesting that multilingual training remains more important than combining layers.

Overall, these results strengthen the distinction between discriminative transfer and calibrated score transfer. English-only probes can retain useful cross-lingual ranking information, but this does not automatically translate into useful multilingual routing. Pooled multilingual training provides a more reliable basis for routing because it supports both high expected success and meaningful cost reduction. ETD further shows that this routing signal can be obtained more efficiently when a shared encoder is used before target-model selection. In this sense, the routing results provide an applied validation of the main probing results: reusable multilingual success prediction requires not only transferable signals, but also scores that remain reliable enough to guide downstream decisions.

5.7. Success Prediction in BFCL

The BFCL results investigate whether pre-generation success prediction is specific to the MATH setting. Unlike the main experiments, BFCL is not multilingual and is not used for routing. It instead evaluates a different dataset with a different prompt format, output

format, and correctness criterion.

The results show that success is linearly decodable in BFCL when supervision is available. Both the BFCL-only probe and the joint MATH+BFCL probe achieve strong AUROC trajectories, indicating that pre-generation activations contain information about whether a function-calling response is likely to be correct. This suggests that success prediction is not limited to mathematical reasoning.

The main finding of the BFCL experiment is that the same probing approach remains effective when examples from the target dataset are available during training. This supports the broader relevance of pre-generation success prediction, while keeping the main conclusions of the thesis centred on multilingual transfer, calibration, and routing in the MATH setting.

5.8. Limitations

This thesis has several limitations that should be considered when interpreting the results. The first limitation concerns task coverage. The experiments evaluate pre-generation success prediction on mathematical reasoning and function calling, which provide two structured settings with automatic correctness evaluation. However, both tasks have relatively well-defined success criteria and do not cover the full range of LLM use cases, such as open-ended dialogue, factual question answering, summarisation, creative writing, or long-form instruction following. The findings therefore show that success prediction can be effective in the evaluated structured tasks, but they should not be assumed to generalise unchanged to all domains or interaction settings.

A second limitation concerns language coverage. The benchmark includes ten languages with different scripts and levels of model competence, but it remains only a subset of the multilingual deployment space. Languages with lower resource availability, different writing systems, or weaker representation in model pre-training may produce different transfer and calibration behaviour. The results also depend on the quality of the translated problem statements. Although the benchmark is designed so that each translated prompt preserves the same mathematical problem, translation can still affect phrasing, ambiguity, tokenisation, and prompt length. These factors may change the apparent difficulty of a problem independently of the underlying mathematical content.

The correctness labels are another source of limitation. Each empirical success score is estimated from five sampled generations, so the target value $y = k/5$ is an estimate of expected success under a specific decoding protocol rather than an exact probability of

correctness. More rollouts would provide a more stable estimate of model success, but would also substantially increase the cost of data collection. The evaluation is therefore tied to the chosen generation settings, including the sampling parameters, prompt format, and correctness extraction procedure.

The probing results should also be interpreted as evidence of linear decodability rather than causal evidence about model computation. A successful probe shows that information about future success is accessible from the hidden representation, but it does not show that the model itself uses the same information when generating an answer. Similarly, the layer-wise results show where success information is most accessible to a linear probe, but they do not by themselves identify the mechanism that produces the observed depth-dependent transfer pattern.

The routing experiment should be interpreted as a simulation of model-selection utility rather than as a complete deployment benchmark. Routing decisions are evaluated using already-generated rollouts and empirical success scores, not by deploying the router in an online serving system. The cost model accounts for prompt processing and answer generation under fixed input and output token prices, but it remains a simplified approximation. It does not account for model batching, hardware utilisation, memory constraints, latency, cache reuse, or scheduling effects. The reported savings therefore measure expected savings under the defined simulation and cost model, rather than exact monetary or latency savings in production.

Finally, the routing results depend on the evaluated candidate model set and on validation-selected routing policies. The conclusions compare English-only transfer and pooled multilingual routing for the specific models included in this thesis, with `gpt-oss-high` used as the strongest-model baseline. Different candidate pools, stronger or weaker anchors, alternative cost assumptions, or different layer-selection procedures could change the resulting accuracy–cost frontier. The routing results therefore provide an applied validation of the probe signal within the evaluated setup, rather than a universal claim about all multilingual routing systems.

6 | Conclusion

This chapter summarises the findings of the thesis, answers the research questions, and outlines the resulting contributions and future research directions.

6.1. Summary of Findings

This thesis studied whether pre-generation success probes in multilingual large language models transfer across languages and whether they can support cost-aware model routing. The central motivation was that collecting labelled rollout data for every language and model is expensive, while always using the strongest available model can make multilingual inference unnecessarily costly. The thesis therefore examined whether success signals learned from hidden activations can be reused across languages and whether these signals remain reliable enough to guide downstream decisions.

The results show that future success is linearly decodable from pre-generation hidden activations across the evaluated models and languages. In the same-language setting, ridge-regression probes achieve consistently informative AUROC values, showing that model correctness is partly reflected in the prompt-conditioned representation before answer generation begins. This supports the premise that pre-generation activations contain information about the model's likelihood of solving a problem.

The cross-lingual results show that this success signal is partly transferable across languages. Probes trained in one language remain informative when evaluated on other languages, although their performance is lower than in the same-language setting. This indicates that the probes do not rely only on language-specific surface patterns, but also capture a component of task difficulty and model competence that is shared across translated versions of the same mathematical problem.

The layer-wise analysis shows that transfer is depth-dependent. Same-language success prediction generally remains strong across the middle-to-late layers, whereas cross-lingual transfer tends to peak before the final layers and then decline. This suggests that the most transferable success-related information is most accessible in intermediate or

late-intermediate representations, rather than necessarily in the final hidden state. The routing-by-depth analysis further shows that this layer-wise structure also affects downstream model selection, especially for English-only transfer.

The calibration results show that discriminative transfer is not sufficient for reliable multilingual success prediction. English-trained probes can rank examples across languages, but their predicted scores are often poorly calibrated under transfer. Pooled multilingual training produces more reliable score scales across target languages, reducing calibration error and making the probe outputs more suitable for threshold-based decisions.

The routing experiment provides an applied validation of these findings. English-only transfer remains too weak to provide a useful standalone multilingual router. By contrast, pooled multilingual routing achieves the strongest accuracy–cost trade-off in the evaluated setting. It reaches 87.63% expected success at its highest-success operating point while achieving a cost saving of 4.83%, and it reaches 86.06% expected success at the anchor-matching operating point with a cost saving of 25.97%. These results show that multilingual probe training can support meaningful cost-aware routing, provided that the probe scores are reliable enough for cross-language and cross-model decision-making.

Finally, the BFCL evaluation shows that the same probing approach can be applied outside MATH when supervision from the target dataset is available. BFCL-only and joint MATH+BFCL probes produce strong success-prediction results on the function-calling benchmark, showing that pre-generation activations also contain success-relevant information in this setting. This result broadens the empirical support for pre-generation success prediction, while the main conclusions of the thesis remain centred on multilingual transfer, calibration, and routing in the MATH setting.

6.2. Answers to Research Questions

RQ1. To what extent do pre-generation success probes transfer across languages?

> The experiments indicate that pre-generation success probes transfer discriminatively across languages. Cross-lingual probes remain informative when trained on one language and evaluated on another, and their AUROC values are consistently above chance across the evaluated models. However, transfer is not lossless. Same-language probes generally outperform cross-lingual probes, showing that part of the success signal is language-specific or affected by language-dependent activation shifts. The answer to RQ1 is therefore that success probes do transfer across languages, but with a measurable reduction relative to same-language prediction.

RQ2. At which model depth are cross-lingual success signals most transferable?

The layer-wise results show that cross-lingual success signals are most transferable in the middle-to-late part of the model, before the final layers. Across the evaluated models, transfer AUROC generally increases after the early layers, reaches its strongest region before the end of the network, and then often declines toward the final layer. This differs from the same-language setting, where performance is more likely to remain strong later in the model. The answer to RQ2 is therefore that transferable success information is depth-dependent and is most accessible before the final generation-facing representations.

RQ3. To what extent can multilingual pre-generation success probes support cost-aware model routing?

The routing results indicate that pre-generation success probes can support cost-aware routing in the evaluated simulation, but their usefulness depends on the reliability of the probe scores used for decision-making and on the cost of obtaining those scores. English-only transfer probes are too weak to provide a useful standalone multilingual routing policy in the evaluated setting. They retain some cross-lingual ranking information, but their score scale is not reliable enough to support strong decisions among candidate models. Pooled multilingual probes provide a more effective routing signal. In the encoder-target decoupled setting, pooled multilingual routing reaches 87.63% expected success at its highest-success operating point while achieving a cost saving of 4.83%, and reaches 86.06% expected success at the anchor-matching operating point with a cost saving of 25.97%. The answer to RQ3 is therefore that multilingual success probes can support cost-aware routing when their scores are sufficiently reliable for cross-language decision-making. In this thesis, pooled multilingual training provides that reliability more effectively than English-only transfer within the evaluated set of languages and models.

6.3. Contributions

This thesis contributes a multilingual empirical study of pre-generation success prediction in large language models. It extends success probing from same-language prediction to a cross-lingual setting, where the same underlying mathematical problems are evaluated across translated prompts. This design makes it possible to separate problem-level difficulty from language-dependent effects more directly than in a setting where each language uses different examples. The evaluation further shows that the same probing framework can also be applied to a substantially different task when supervision is available.

A second contribution is the layer-wise analysis of cross-lingual success prediction. By training and evaluating probes across model depth, the thesis identifies where success-related information is most accessible and where it transfers most effectively across languages. The results show that the strongest transferable signal is not necessarily located at the final layer, which is important for both representation analysis and downstream use of probes.

A third contribution is the distinction between discriminative transfer and calibrated score transfer. The thesis shows that English-trained probes can retain useful ranking information across languages, but that their scores are not always reliable as numerical success estimates. Pooled multilingual training improves this reliability, showing that multilingual success prediction requires attention to both AUROC and calibration.

A final contribution is the routing case study. The routing simulation connects probe quality to an applied inference problem by evaluating whether success estimates can guide model selection under cost constraints. The results show that pooled multilingual probes can produce meaningful cost savings while preserving expected success, whereas English-only transfer does not provide a useful routing policy in the evaluated setup.

6.4. Future Work

Several directions remain open for future work. One important direction is broader task coverage. This thesis evaluates pre-generation success prediction on mathematical reasoning and function calling, both of which are structured tasks with automatic correctness evaluation. Future work could study whether similar success signals appear in less constrained settings, such as factual question answering, summarisation, open-ended dialogue, code generation, or multi-turn instruction following. This would help determine how far success probing generalises beyond tasks with clearly defined output formats and correctness criteria.

Future work could also expand the multilingual setting. The present thesis evaluates ten languages, but broader coverage would help determine how success probes behave for lower-resource languages, languages with different scripts, and languages that are less strongly represented in model pre-training. This would make it possible to study whether transfer depends more strongly on linguistic similarity, script, tokenisation, translation quality, or model-specific language competence.

Calibration also deserves deeper analysis. Expected Calibration Error and reliability diagrams provide useful aggregate summaries, but future work could analyse calibration

across problem difficulty, language pairs, model families, and confidence regions. It could also evaluate different calibration methods or language-aware calibration strategies to determine whether transferred probe scores can be made more reliable without requiring full target-language supervision.

The routing experiment could be extended toward a more realistic deployment setting. The routing results in this thesis are based on a simulation over already-generated rollouts and use a simplified cost model. A more complete evaluation could include latency, batching, hardware utilisation, provider-specific prices, model availability, and the runtime cost of feature extraction. It could also evaluate online routing, where thresholds are updated as new data become available and where the cost of obtaining pre-generation activations is measured directly.

Finally, future work could study richer probe designs while preserving the interpretability of the current approach. This thesis uses linear ridge-regression probes to test whether success information is linearly accessible from hidden activations. More expressive probes, task-conditioned probes, or lightweight multi-task probes may improve prediction performance, but they would also require careful controls to distinguish information encoded in the representation from information learned by the probe itself. Such work could clarify the trade-off between predictive performance, interpretability, and practical usefulness for routing.

Bibliography

Agarwal, A., Jian, J., Manning, C. D., & Murty, S. (2025). Mechanisms vs. outcomes: Probing for syntax fails to explain performance on targeted syntactic evaluations. *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, 33737–33757. https://doi.org/10.18653/v1/2025.emnlp-main.1712

Agarwal, S., Ahmad, L., Ai, J., Altman, S., Applebaum, A., Arbus, E., Arora, R. K., Bai, Y., Baker, B., Bao, H., et al. (2025). gpt-oss-120b and gpt-oss-20b model card.

Alain, G., & Bengio, Y. (2016). Understanding intermediate layers using linear classifier probes. https://doi.org/10.48550/arXiv.1610.01644

Anthropic. (2026). Claude pricing [Pricing page for Claude API models. Accessed: 2026-06-20.]. https://claude.com/pricing

Azaria, A., & Mitchell, T. (2023). The internal state of an llm knows when it's lying. *Findings of the Association for Computational Linguistics: EMNLP 2023*, 967–976.

Ba, J. L., Kiros, J. R., & Hinton, G. E. (2016). Layer normalization. https://doi.org/10.48550/arXiv.1607.06450

Bandarkar, L., Yang, C., Fayyaz, M., Hu, J., & Peng, N. (2025). Multilingual routing in mixture-of-experts. https://doi.org/10.48550/arXiv.2510.04694

Belinkov, Y. (2022). Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, *48*(1), 207–219. https://doi.org/10.1162/coli_a_00422

Bengio, Y., Ducharme, R., Vincent, P., & Jauvin, C. (2003). A neural probabilistic language model. *Journal of Machine Learning Research*, *3*, 1137–1155.

Blevins, T., Gonen, H., & Zettlemoyer, L. (2022). Analyzing the mono- and cross-lingual pretraining dynamics of multilingual language models. *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, 3575–3590. https://doi.org/10.18653/v1/2022.emnlp-main.234

Blevins, T., Limisiewicz, T., Gururangan, S., Li, M., Gonen, H., Smith, N. A., & Zettlemoyer, L. (2024). Breaking the curse of multilinguality with cross-lingual expert language models. *Proceedings of the 2024 Conference on Empirical Methods in Nat-*

ural Language Processing, 10822–10837. https://doi.org/10.18653/v1/2024.emnlp-main.604

Bradley, A. P. (1997). The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern Recognition*, *30*(7), 1145–1159. https://doi.org/10.1016/S0031-3203(96)00142-2

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *Advances in Neural Information Processing Systems (NeurIPS)*, *33*, 1877–1901.

Chen, L., Zaharia, M., & Zou, J. (2023). FrugalGPT: How to use large language models while reducing cost and improving performance. https://doi.org/10.48550/arXiv.2305.05176

Civelli, S., Bernardelle, P., Brunello, N., & Demartini, G. (2026). A shared geometry of difficulty in multilingual language models. https://doi.org/10.48550/arXiv.2601.12731

Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, E., Ott, M., Zettlemoyer, L., & Stoyanov, V. (2020). Unsupervised cross-lingual representation learning at scale. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 8440–8451. https://doi.org/10.18653/v1/2020.acl-main.747

De Varda, A. G., & Marelli, M. (2023). Data-driven cross-lingual syntax: An agreement study with massively multilingual models. *Computational Linguistics*, *49*(2), 261–299.

DeepL SE. (2026). Deepl translator [Accessed: 2026-04-29]. https://www.deepl.com/translator

Dekoninck, J., Baader, M., & Vechev, M. (2025). A unified approach to routing and cascading for llms. *Proceedings of the 42nd International Conference on Machine Learning*, 12987–13010.

Dufter, P., Schmitt, M., & Schütze, H. (2022). Position information in transformers: An overview. *Computational Linguistics*, *48*(3), 733–763.

Elazar, Y., Ravfogel, S., Jacovi, A., & Goldberg, Y. (2021). Amnesic probing: Behavioral explanation with amnesic counterfactuals. *Transactions of the Association for Computational Linguistics*, *9*, 160–175.

Fang, M., Wan, X., Lu, F., Xing, F., & Zou, K. (2025). Mathodyssey: Benchmarking mathematical problem-solving skills in large language models using odyssey math data. *Scientific data*, *12*(1), 1392.

Fawcett, T. (2006). An introduction to roc analysis. *Pattern Recognition Letters*, *27*(8), 861–874. https://doi.org/10.1016/j.patrec.2005.10.010

Google. (2026). Gemini developer api pricing [Pricing page for Google Gemini API models. Accessed: 2026-06-20.]. https://ai.google.dev/gemini-api/docs/pricing

Guo, C., Pleiss, G., Sun, Y., & Weinberger, K. Q. (2017). On calibration of modern neural networks. *Proceedings of the 34th International Conference on Machine Learning*, *70*, 1321–1330. https://proceedings.mlr.press/v70/guo17a.html

Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., & Steinhardt, J. (2021). Measuring mathematical problem solving with the math dataset. *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*.

Hewitt, J., & Liang, P. (2019). Designing and interpreting probes with control tasks. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, 2733–2743. https://doi.org/10.18653/v1/D19-1275

Holtzman, A., Buys, J., Du, L., Forbes, M., & Choi, Y. (2020). The curious case of neural text degeneration. *International Conference on Learning Representations*. https://openreview.net/forum?id=rygGQyrFvH

Jawahar, G., Sagot, B., & Seddah, D. (2019). What does BERT learn about the structure of language? *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 3651–3657. https://doi.org/10.18653/v1/P19-1356

Jiang, Z., Liu, A., & Van Durme, B. (2022). Calibrating zero-shot cross-lingual (un-)structured predictions. *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, 2648–2674. https://doi.org/10.18653/v1/2022.emnlp-main.170

Jitkrittum, W., Narasimhan, H., Rawat, A. S., Juneja, J., Wang, C., Wang, Z., Go, A., Lee, C.-Y., Shenoy, P., Panigrahy, R., Menon, A. K., & Kumar, S. (2025). Universal model routing for efficient llm inference. https://doi.org/10.48550/arXiv.2502.08773

Kasnavieh, H. H., Haffari, G., Leckie, C., & Toosi, A. N. (2026). Introlm: Introspective language models via prefilling-time self-evaluation.

Kudo, T., & Richardson, J. (2018). SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 66–71. https://doi.org/10.18653/v1/D18-2012

Libovickỳ, J., Rosa, R., & Fraser, A. (2020). On the language neutrality of pre-trained multilingual representations. *Findings of the Association for Computational Lin-*

guistics: EMNLP 2020, 1663–1674. https://doi.org/10.18653/v1/2020.findings-emnlp.150

Limisiewicz, T., Balhar, J., & Mareček, D. (2023). Tokenization impacts multilingual language modeling: Assessing vocabulary allocation and overlap across languages. *Findings of the Association for Computational Linguistics: ACL 2023*, 5661–5681. https://doi.org/10.18653/v1/2023.findings-acl.350

Lin, S., Hilton, J., & Evans, O. (2022). Teaching models to express their uncertainty in words.

Lugoloobi, W., Foster, T., Bankes, W., & Russell, C. (2026). LLMs encode their failures: Predicting success from pre-generation activations. https://doi.org/10.48550/arXiv.2602.09924

Lugoloobi, W., & Russell, C. (2025). LLMs encode how difficult problems are.

Lupaşcu, M., Rogoz, A.-C., Sorin Stupariu, M., & Tudor Ionescu, R. (2026). Large multimodal models for low-resource languages: A survey. *Information Fusion*, *131*, 104189. https://doi.org/10.1016/j.inffus.2026.104189

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *Advances in Neural Information Processing Systems*, *26*.

Ong, I., Almahairi, A., Wu, V., Chiang, W.-L., Wu, T., Gonzalez, J. E., Kadous, M. W., & Stoica, I. (2024). Routellm: Learning to route llms with preference data. https://doi.org/10.48550/arXiv.2406.18665

OpenAI. (2026). Openai api pricing [Pricing page for OpenAI API models. Accessed: 2026-06-20.]. https://platform.openai.com/docs/pricing/

Patil, S. G., Mao, H., Yan, F., Ji, C. C.-J., Suresh, V., Stoica, I., & Gonzalez, J. E. (2025). The Berkeley Function Calling Leaderboard (BFCL): From tool use to agentic evaluation of large language models. *Proceedings of the 42nd International Conference on Machine Learning*, *267*, 48371–48392.

Petrov, A., La Malfa, E., Torr, P. H. S., & Bibi, A. (2023). Language model tokenizers introduce unfairness between languages. *Advances in Neural Information Processing Systems*, *36*, 36963–36990.

Pires, T., Schlinger, E., & Garrette, D. (2019). How multilingual is multilingual BERT? *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 4996–5001. https://doi.org/10.18653/v1/P19-1493

Qin, L., Chen, Q., Zhou, Y., Chen, Z., Li, Y., Liao, L., Li, M., Che, W., & Yu, P. S. (2025). A survey of multilingual large language models. *Patterns*, *6*(1).

Qwen Team. (2026). Qwen3.5 model collection. https://huggingface.co/collections/Qwen/qwen35

Rei, R., Treviso, M., Guerreiro, N. M., Zerva, C., Farinha, A. C., Maroti, C., de Souza, J. G. C., Glushkova, T., Alves, D. M., Lavie, A., Coheur, L., & Martins, A. F. T. (2022). CometKiwi: IST-Unbabel 2022 submission for the quality estimation shared task. *Proceedings of the Seventh Conference on Machine Translation*.

Rust, P., Pfeiffer, J., Vulić, I., Ruder, S., & Gurevych, I. (2021). How good is your tokenizer? on the monolingual performance of multilingual language models. *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 3118–3135. https://doi.org/10.18653/v1/2021.acl-long.243

Scarlatos, A., Fernandez, N., Ormerod, C., Lottridge, S., & Lan, A. (2025). SMART: Simulated Students Aligned with Item Response Theory for Question Difficulty Prediction. *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, 25071–25094. https://doi.org/10.18653/v1/2025.emnlp-main.1274

Sennrich, R., Haddow, B., & Birch, A. (2016). Neural machine translation of rare words with subword units. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, 1715–1725. https://doi.org/10.18653/v1/P16-1162

Song, J., Xu, Z., & Zhong, Y. (2025). Out-of-distribution generalization via composition: A lens through induction heads in Transformers. *Proceedings of the National Academy of Sciences*, *122*(6), e2417182122. https://doi.org/10.1073/pnas.2417182122

Song, W., Huang, Z., Cheng, C., Gao, W., Xu, B., Zhao, G., Wang, F., & Wu, R. (2025). Irt-router: Effective and interpretable multi-llm routing via item response theory. *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 15629–15644. https://doi.org/10.18653/v1/2025.acl-long.761

Sun, Y., Stolfo, A., & Sachan, M. (2025). Probing for arithmetic errors in language models. *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, 8122–8139.

Tenney, I., Das, D., & Pavlick, E. (2019). BERT rediscovers the classical NLP pipeline. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 4593–4601. https://doi.org/10.18653/v1/P19-1452

Unanue, I. J., Haffari, G., & Piccardi, M. (2023). T3l: Translate-and-test transfer learning for cross-lingual text classification. *Transactions of the Association for Computational Linguistics*, *11*, 1147–1161.

Varshney, T., Surla, A., Xu, M., Venkata Krishnan, G., Jeblick, M., Austin, D., Vaidya, N., & Onofrio, D. (2026). LLM Router: Prefill is all you need.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, *30*.

Voita, E., & Titov, I. (2020). Information-theoretic probing with minimum description length. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, 183–196. https://doi.org/10.18653/v1/2020.emnlp-main.14

Wu, Y., Wu, S., Tao, Y., Li, Y., & Sarwate, A. D. (2025). From deferral to learning: Online in-context knowledge distillation for llm cascades. https://doi.org/10.48550/arXiv.2509.22984

Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., Zheng, C., Liu, D., Zhou, F., Huang, F., Hu, F., Ge, H., Wei, H., Lin, H., Tang, J., … Qiu, Z. (2025). Qwen3 technical report.

Yeo, S., Ma, Y.-S., Kim, S. C., Jun, H., & Kim, T. (2024). Framework for evaluating code generation ability of large language models. *ETRI Journal*, *46*(1), 106–117. https://doi.org/10.4218/etrij.2023-0357

A | Prompt Templates

This appendix reports the language-specific prompt components used during answer generation. As described in Section 3.3.1, each prompt is formatted as a two-message chat consisting of a system message and a user message. Table A.1 lists the system prompt and suffix used for each evaluated language.

Language	System prompt	User suffix
English	Please reason step by step, and put your final answer within `\boxed{}`.	Let's think step by step and output the final answer within `\boxed{}`.
French	Veuillez raisonner étape par étape et mettre votre réponse finale dans `\boxed{}`.	Réfléchissons étape par étape et indiquons la réponse finale dans `\boxed{}`.
Italian	Per favore, ragiona passo per passo e inserisci la risposta finale dentro `\boxed{}`.	Pensiamo passo per passo e scriviamo la risposta finale dentro `\boxed{}`.
Russian	Пожалуйста, рассуждайте шаг за шагом и поместите окончательный ответ в `\boxed{}`.	Давайте рассуждать шаг за шагом и запишем окончательный ответ в `\boxed{}`.
Chinese	请逐步推理，并将最终答案放在 `\boxed{}` 中。	让我们逐步思考，并将最终答案放在 `\boxed{}` 中。
Arabic	لخاد ةيئاهنلا ةباجإلا عضو، ةوطخب ةوطخ لالدتسالا بجري `\boxed{}`.	لخاد ةيئاهنلا ةباجإلا بتكنو ةوطخب ةوطخ ركفنل `\boxed{}`.
Thai	กรุณาให้เหตุผลทีละขั้นตอน และใส่คำตอบสุดท้ายไว้ใน `\boxed{}`	มาคิดทีละขั้นตอนและแสดงคำตอบสุดท้ายใน `\boxed{}`
Swahili	Tafadhali fikiri hatua kwa hatua, na weka jibu lako la mwisho ndani ya `\boxed{}`.	Hebu tufikirie hatua kwa hatua na kutoa jibu la mwisho ndani ya `\boxed{}`.
Telugu	దయచేసి దశలవారీగా వాదించండి మరియు మీ చివరి సమాధానాన్ని `\boxed{}` లో ఉంచండి.	దశలవారీగా ఆలోచించి చివరి సమాధానాన్ని `\boxed{}` లో రాద్దాం.
Turkish	Lütfen adım adım akıl yürütün ve nihai cevabınızı `\boxed{}` içine yazın.	Adım adım düşünelim ve nihai cevabı `\boxed{}` içinde verelim.

Table A.1: Language-specific prompts used for answer generation. Each problem is formatted as a two-message chat consisting of a system prompt and a user message. The user message contains the translated problem statement followed by the language-specific suffix shown in the table. The full chat is serialised using the model's own chat template before generation.

B | Additional Results

This appendix reports supplementary results that support the main analysis, including additional AUROC trajectories, reliability diagrams, and routing results.

B.1. Same-language AUROC by Language and Model

Table B.1 reports the peak same-language AUROC for each language and evaluated model. Unlike Table 4.2, which reports the peak of the mean trajectory across languages, this table reports the best layer separately for each language and model before averaging across languages. These values therefore summarise the strongest same-language discrimination achieved for each individual language.

Language	**Qwen3 0.6B**	**Qwen3 1.7B**	**Qwen3 4B**	**Qwen3 8B**	**Qwen3.5 4B**	**Qwen3.5 9B**	**gpt-oss low**	**gpt-oss high**	**Mean**
Arabic	0.839	0.802	0.797	0.779	0.797	0.778	0.770	0.766	0.791
Chinese	0.789	0.789	0.744	0.796	0.819	0.734	0.771	0.750	0.774
English	0.832	0.805	0.795	0.770	0.828	0.747	0.748	0.757	0.785
French	0.830	0.795	0.761	0.774	0.768	0.763	0.764	0.821	0.784
Italian	0.822	0.796	0.731	0.766	0.773	0.791	0.764	0.778	0.778
Russian	0.858	0.801	0.766	0.789	0.785	0.833	0.734	0.756	0.790
Swahili	0.884	0.891	0.823	0.791	0.800	0.778	0.706	0.744	0.802
Telugu	0.878	0.824	0.801	0.801	0.854	0.810	0.781	0.760	0.814
Thai	0.854	0.806	0.777	0.787	0.812	0.816	0.717	0.706	0.784
Turkish	0.819	0.819	0.796	0.759	0.781	0.727	0.750	0.747	0.775
Mean	0.840	0.813	0.779	0.781	0.802	0.778	0.750	0.758	0.788

Table B.1: Peak same-language AUROC by language and evaluated model. Each value is the highest AUROC achieved across layers for a probe trained and evaluated on the same language. The final row reports the mean across languages for each model, and the final column reports the mean across models for each language.

B.2. Layer-wise AUROC Trajectories

This appendix reports the complete set of layer-wise AUROC trajectories used in Section 4.2. Each figure contains two panels. The left panel reports same-language AUROC, where probes are trained and evaluated on the same language. The right panel reports cross-lingual transfer AUROC, where probes are trained on one source language and evaluated on a different target language. Curves report mean AUROC with shaded variation across languages.

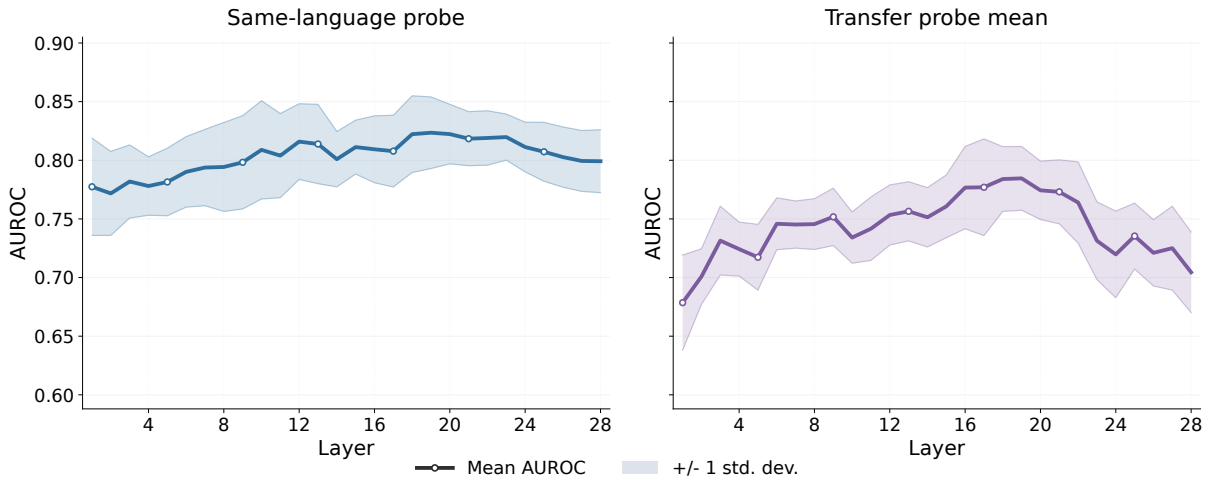


Figure B.1: Layer-wise same-language and cross-lingual transfer AUROC for `Qwen3-0.6B`.

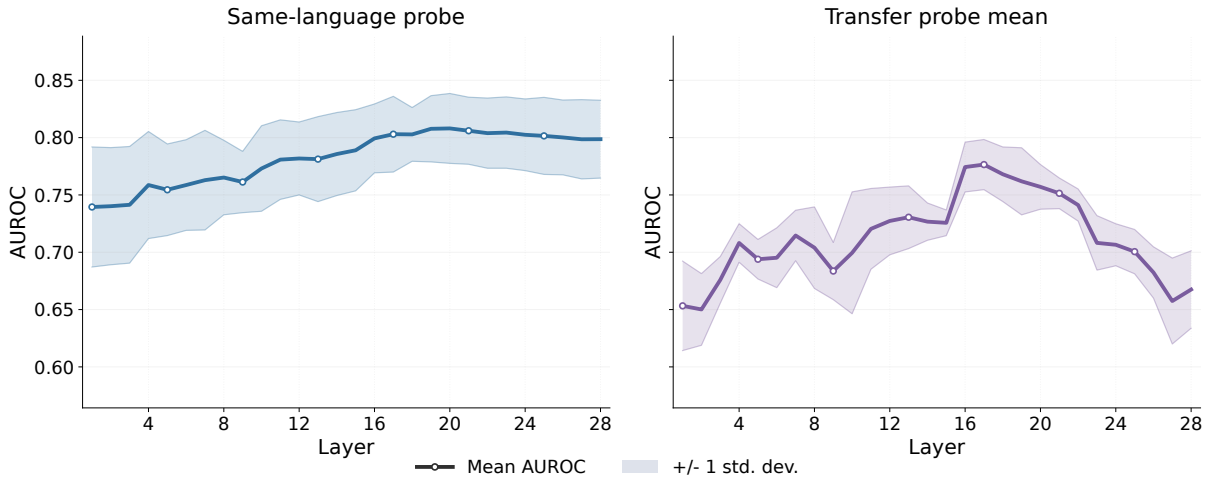


Figure B.2: Layer-wise same-language and cross-lingual transfer AUROC for `Qwen3-1.7B`.

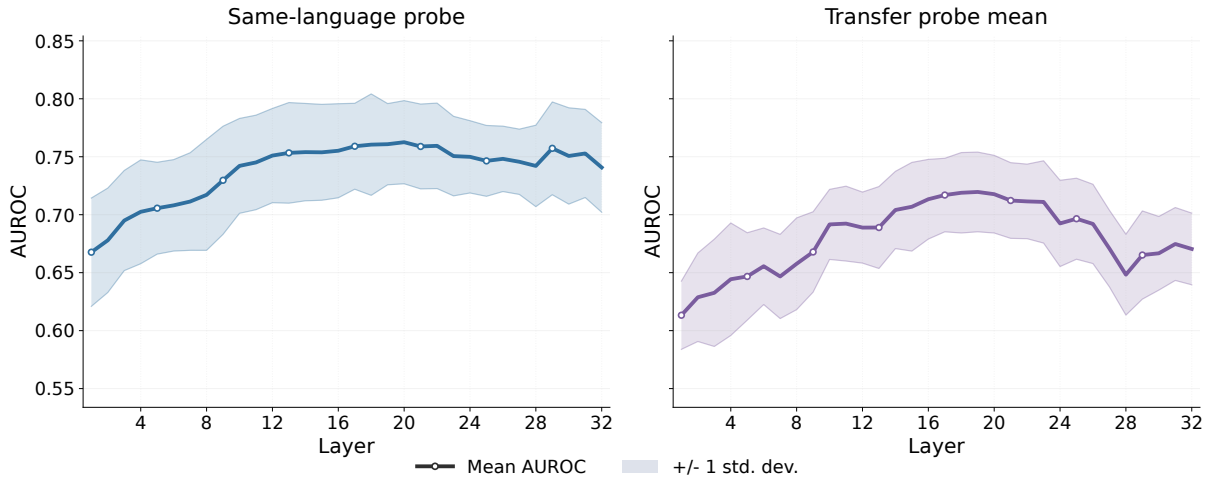


Figure B.3: Layer-wise same-language and cross-lingual transfer AUROC for `Qwen3.5-9B`.

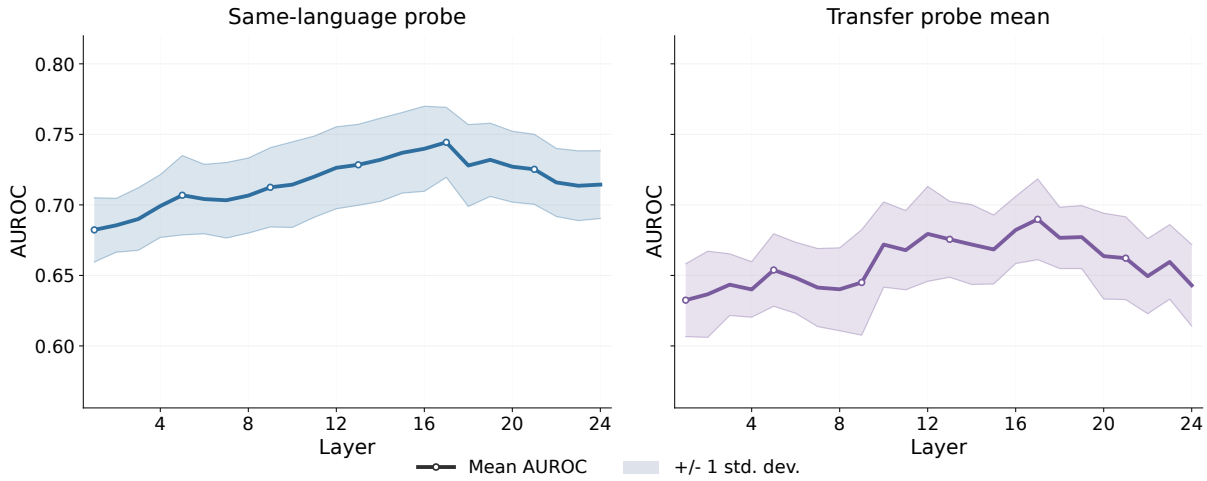


Figure B.4: Layer-wise same-language and cross-lingual transfer AUROC for `gpt-oss-low`.

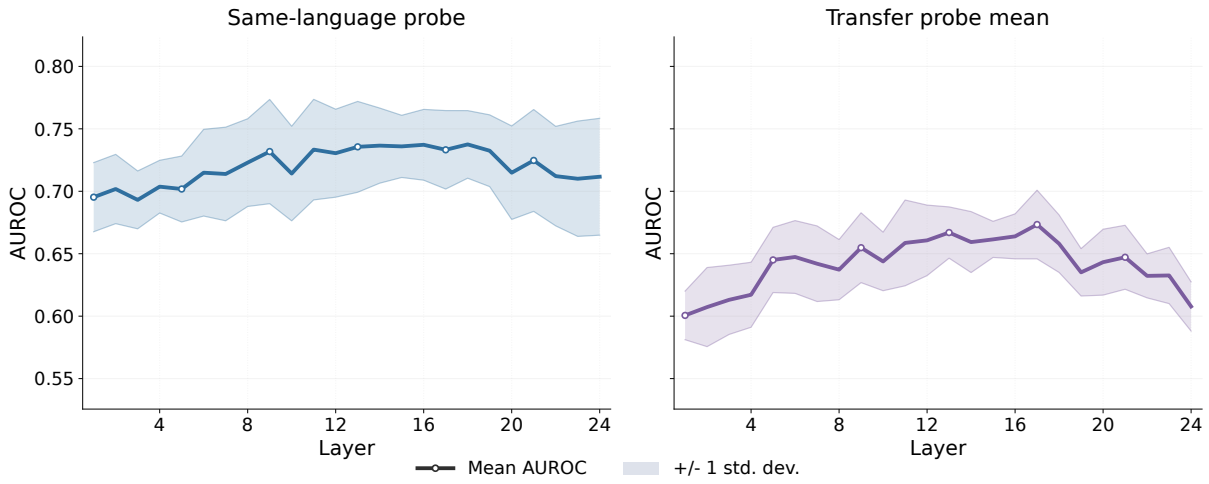


Figure B.5: Layer-wise same-language and cross-lingual transfer AUROC for `gpt-oss-high`.

B.3. Reliability Diagrams

This appendix reports the complete set of reliability diagrams used in Section 4.3. Each figure contains one panel per target language. The diagonal line represents perfect calibration, while deviations from the diagonal indicate that the predicted success scores are either overconfident or underconfident relative to the empirical success rates.

B.3.1. English-trained Probe Reliability

Figures B.6–B.11 report reliability diagrams for English-trained probes evaluated on each target language. In each case, the probe is trained on English and evaluated separately on the ten target languages.

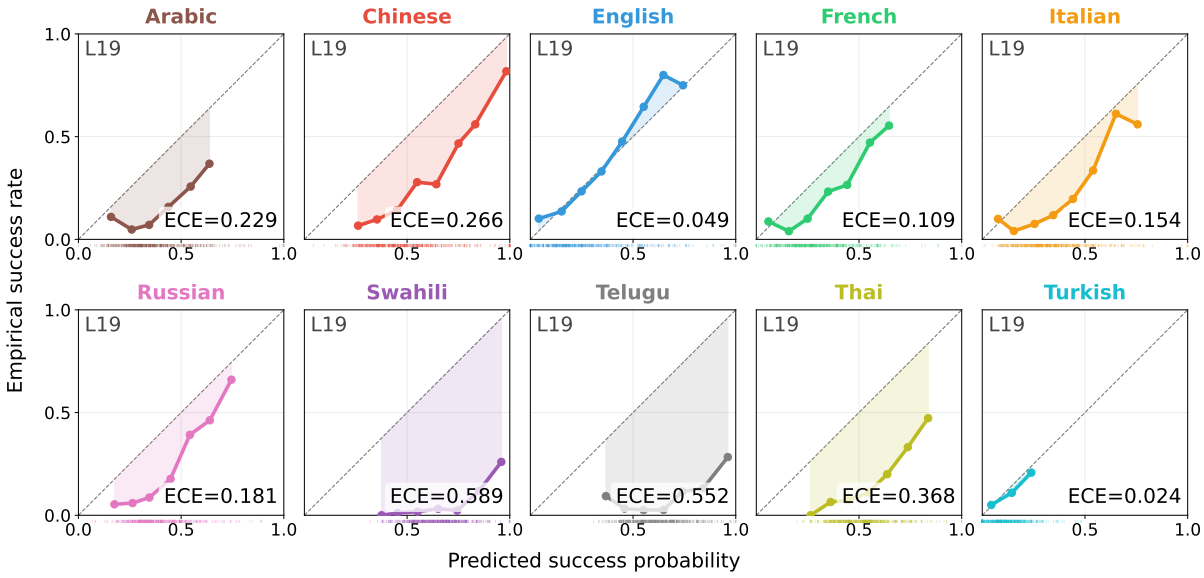


Figure B.6: Reliability diagrams for the English-trained transfer probe on `Qwen3-0.6B`.

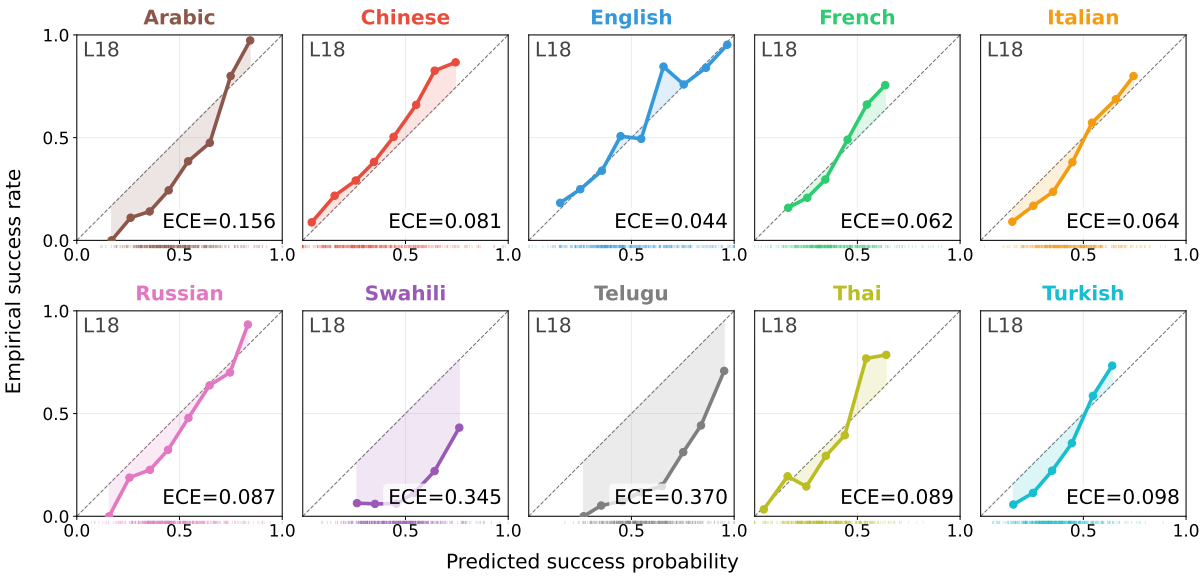


Figure B.7: Reliability diagrams for the English-trained transfer probe on `Qwen3-1.7B`.

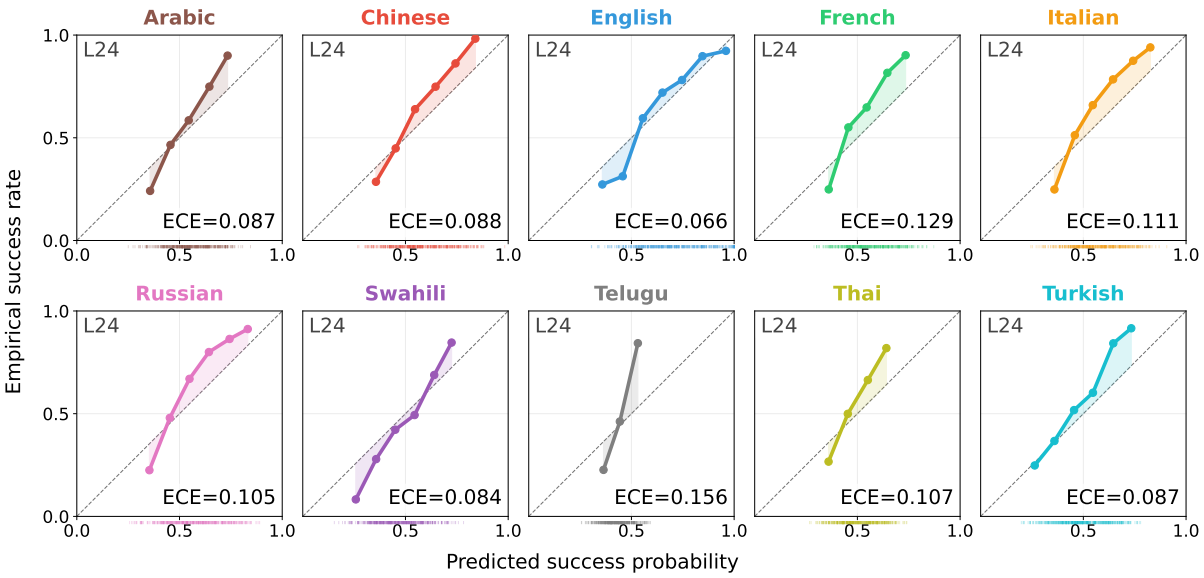


Figure B.8: Reliability diagrams for the English-trained transfer probe on `Qwen3-8B`.

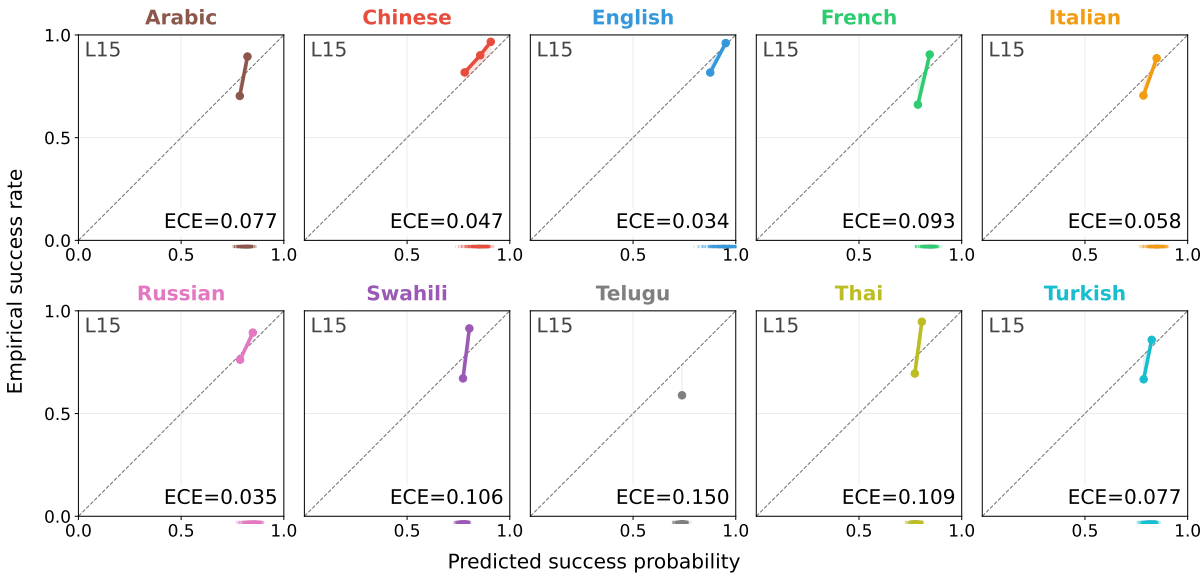


Figure B.9: Reliability diagrams for the English-trained transfer probe on `Qwen3.5-9B`.

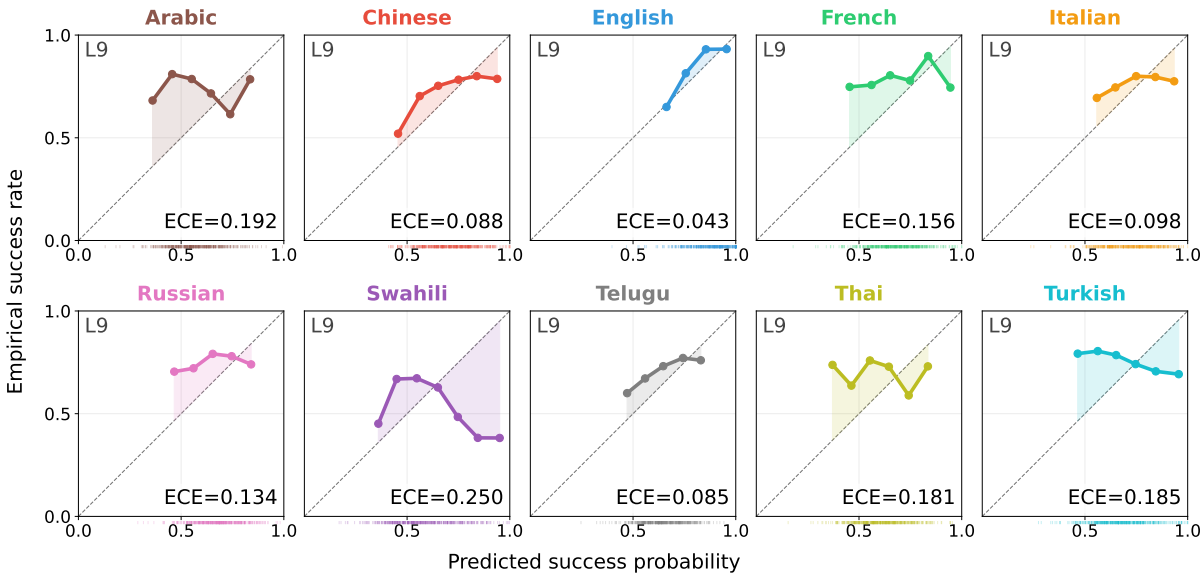


Figure B.10: Reliability diagrams for the English-trained transfer probe on `gpt-oss-low`.

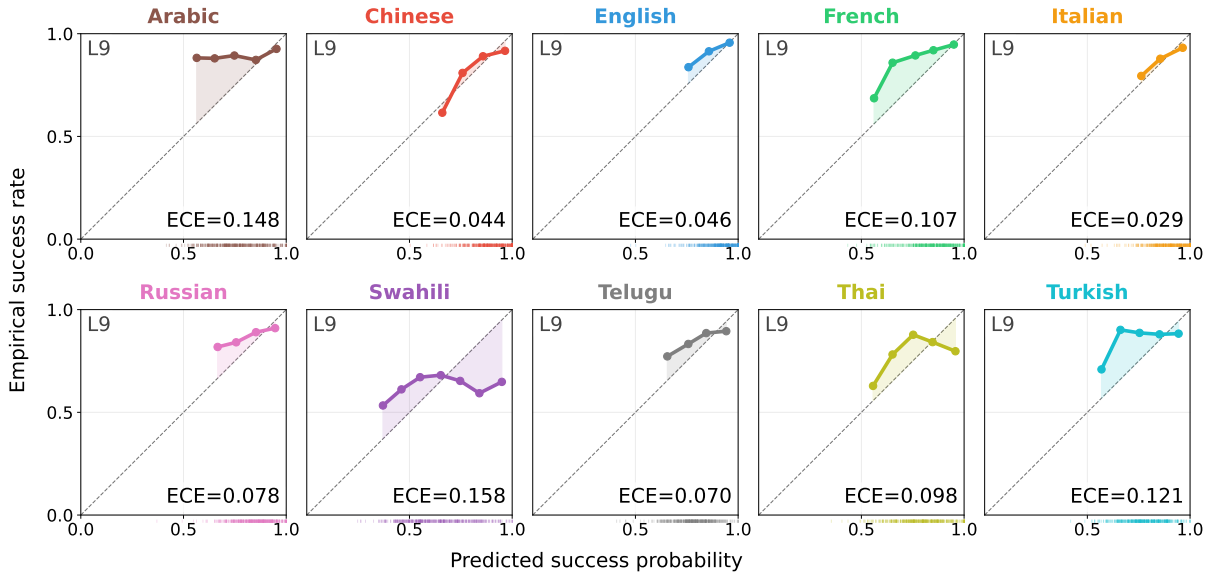


Figure B.11: Reliability diagrams for the English-trained transfer probe on `gpt-oss-high`.

B.3.2. Pooled Multilingual Probe Reliability

Figures B.12–B.17 report reliability diagrams for pooled multilingual probes evaluated on each target language. In each case, the probe is trained using the pooled multilingual configuration described in Section 3.5.4 and evaluated separately on the ten target languages.

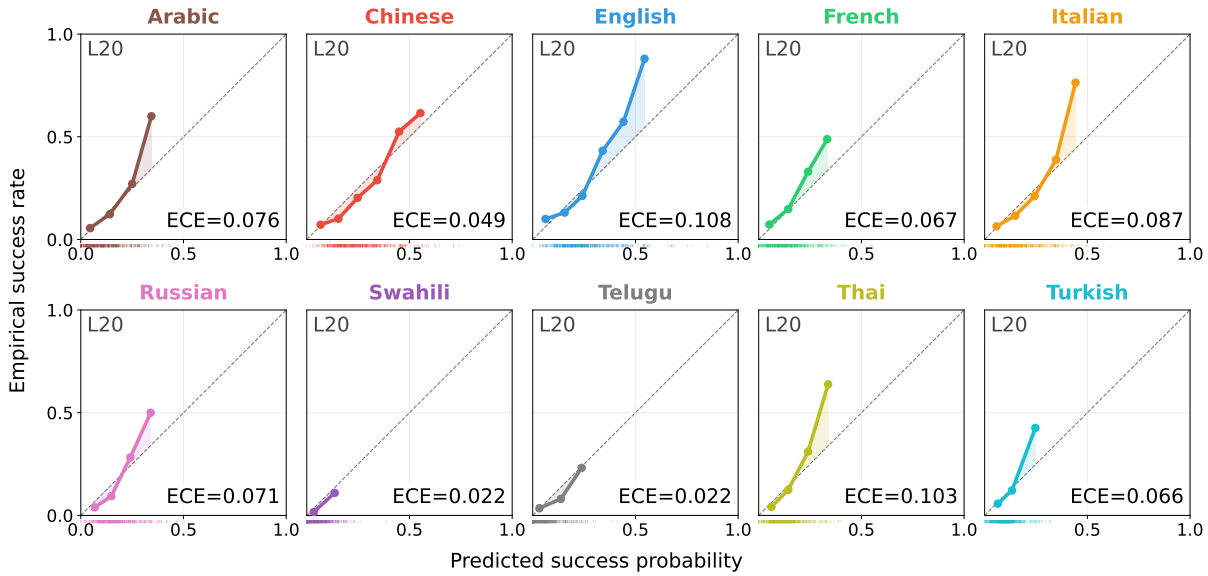


Figure B.12: Reliability diagrams for the pooled multilingual probe on `Qwen3-0.6B`.

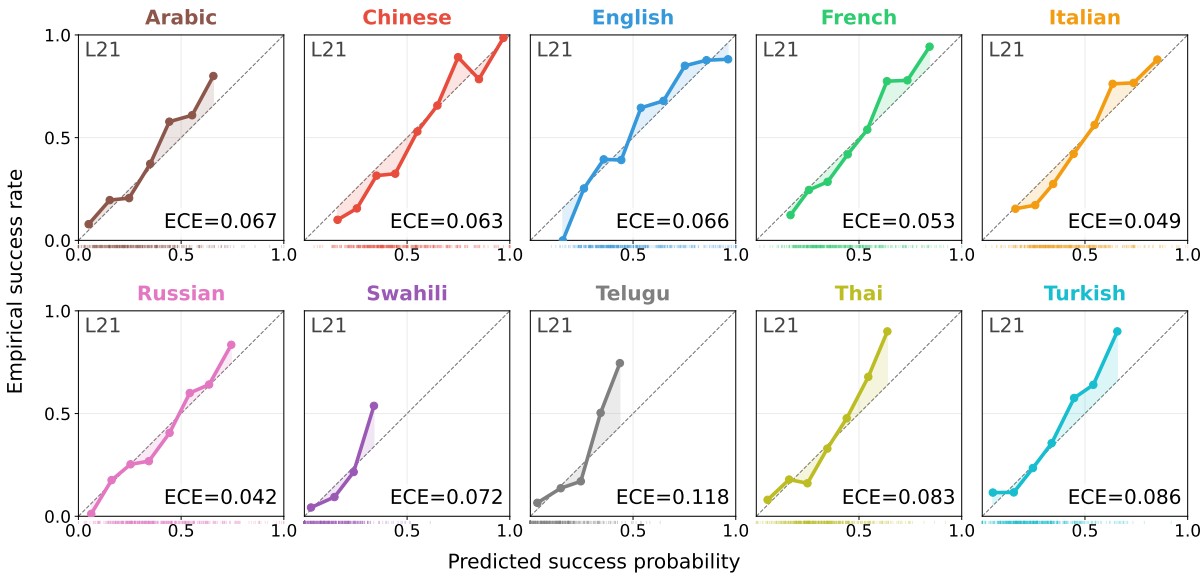


Figure B.13: Reliability diagrams for the pooled multilingual probe on `Qwen3-1.7B`.

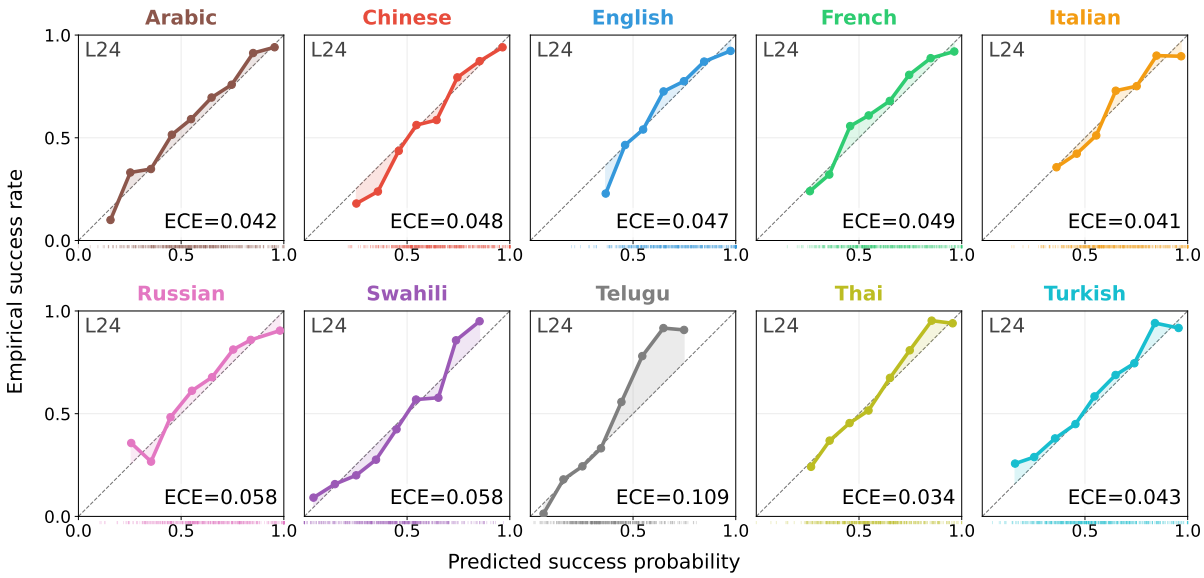


Figure B.14: Reliability diagrams for the pooled multilingual probe on `Qwen3-8B`.

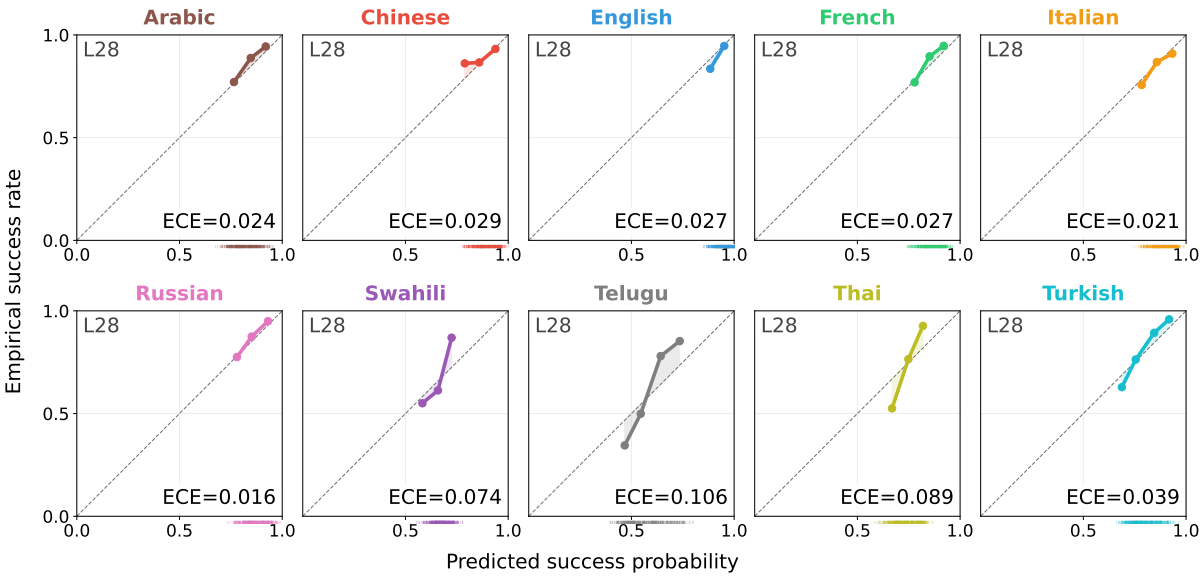


Figure B.15: Reliability diagrams for the pooled multilingual probe on `Qwen3.5-9B`.

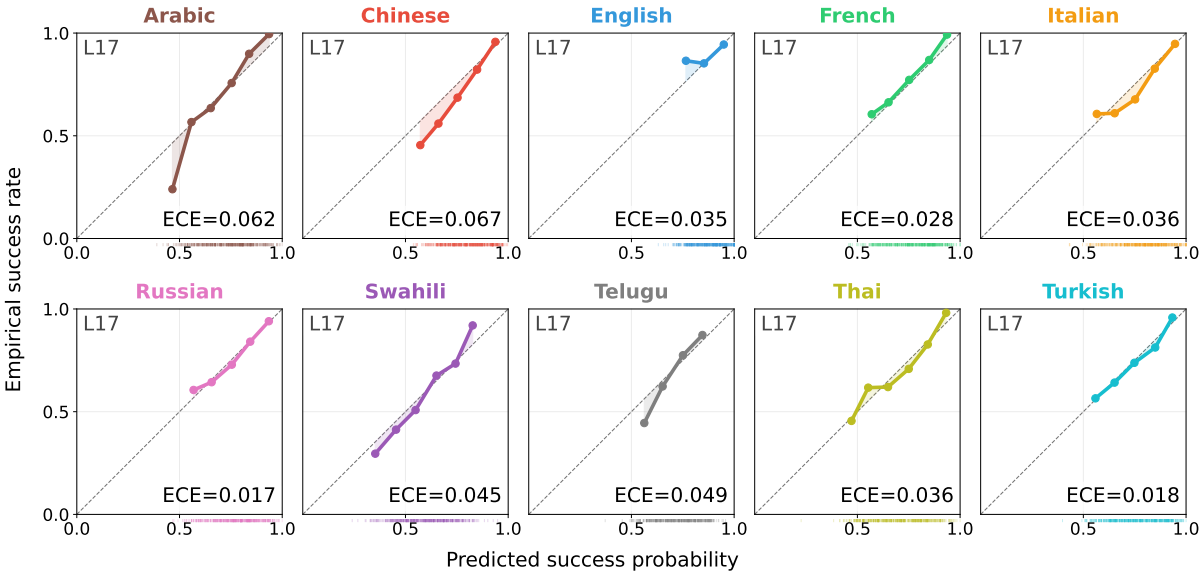


Figure B.16: Reliability diagrams for the pooled multilingual probe on `gpt-oss-low`.

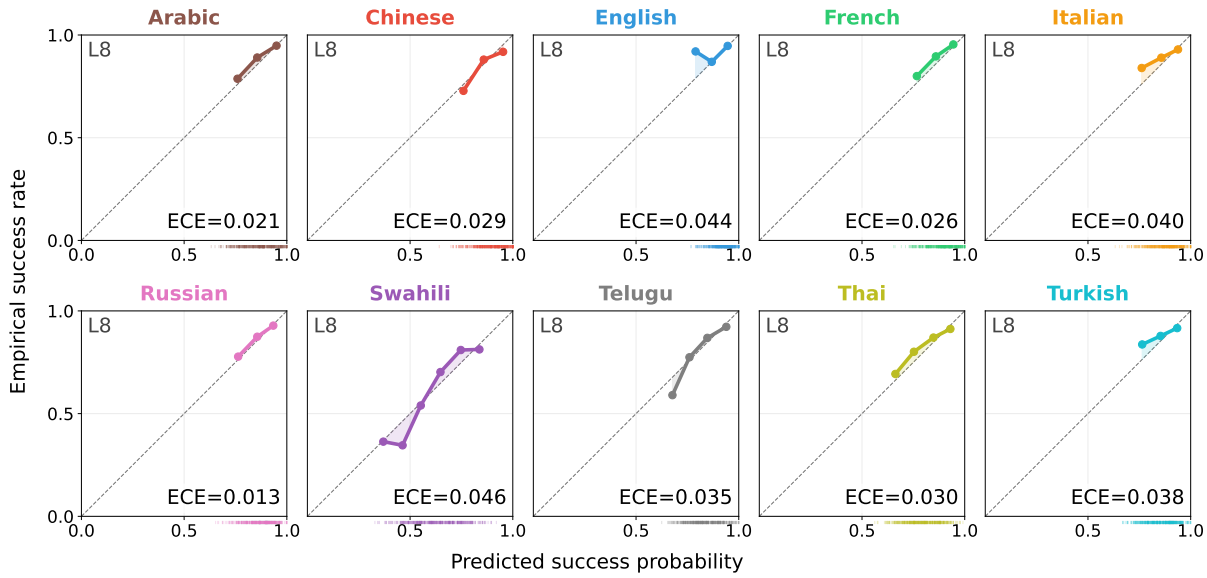


Figure B.17: Reliability diagrams for the pooled multilingual probe on `gpt-oss-high`.

B.4. Additional Routing Results

This appendix reports additional routing results that support the main routing analysis in Section 4.4. The main text focuses on the representative operating points and the comparisons needed to answer the routing research question. The tables and figures below provide the corresponding encoder-selection, model-selection, and layer-ensemble details.

B.4.1. ETD Encoder Selection

Figure B.18 compares the ETD validation frontiers obtained with different encoder models. Each curve uses the best validation-selected layer for the corresponding encoder. The strongest maximum validation success is obtained by the `Qwen3-4B` encoder at layer 23, which is why it is used for the main ETD routing results.

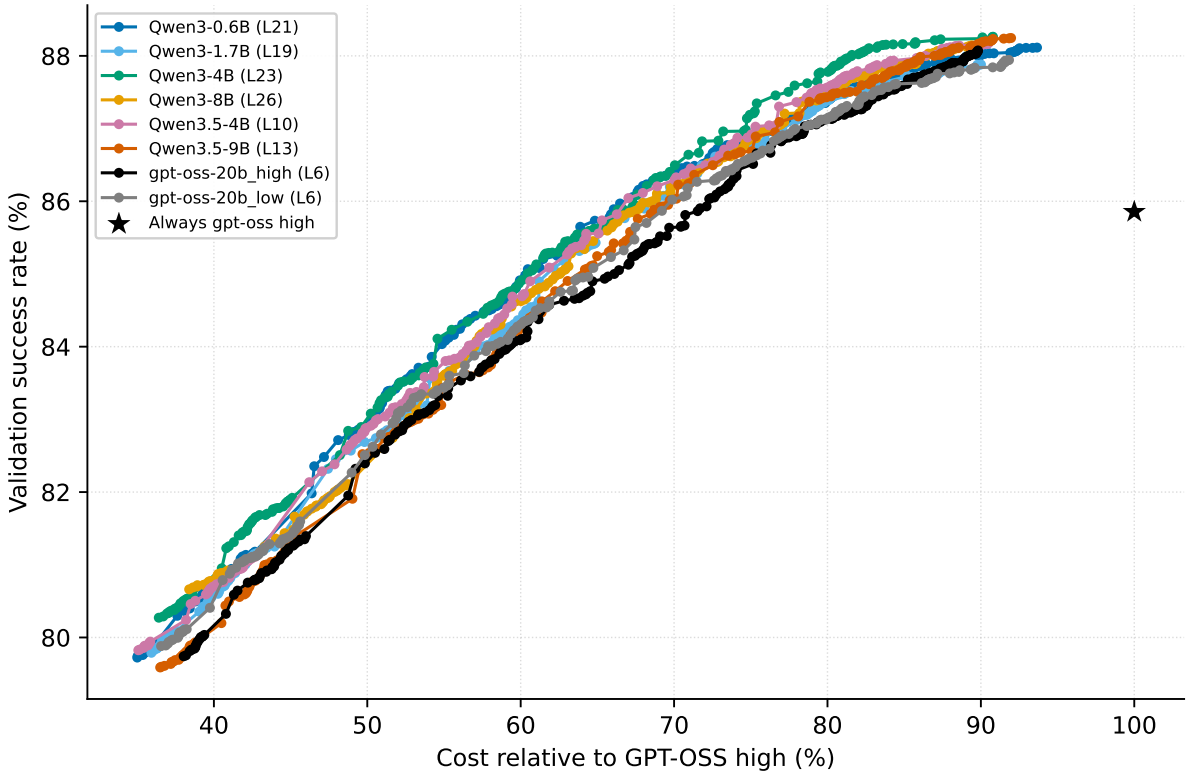


Figure B.18: Validation accuracy–cost frontiers for ETD routing across encoder choices. Each curve corresponds to one encoder model using its best validation-selected layer. Cost is reported as a percentage of the evaluated cost of always selecting `gpt-oss-high`.

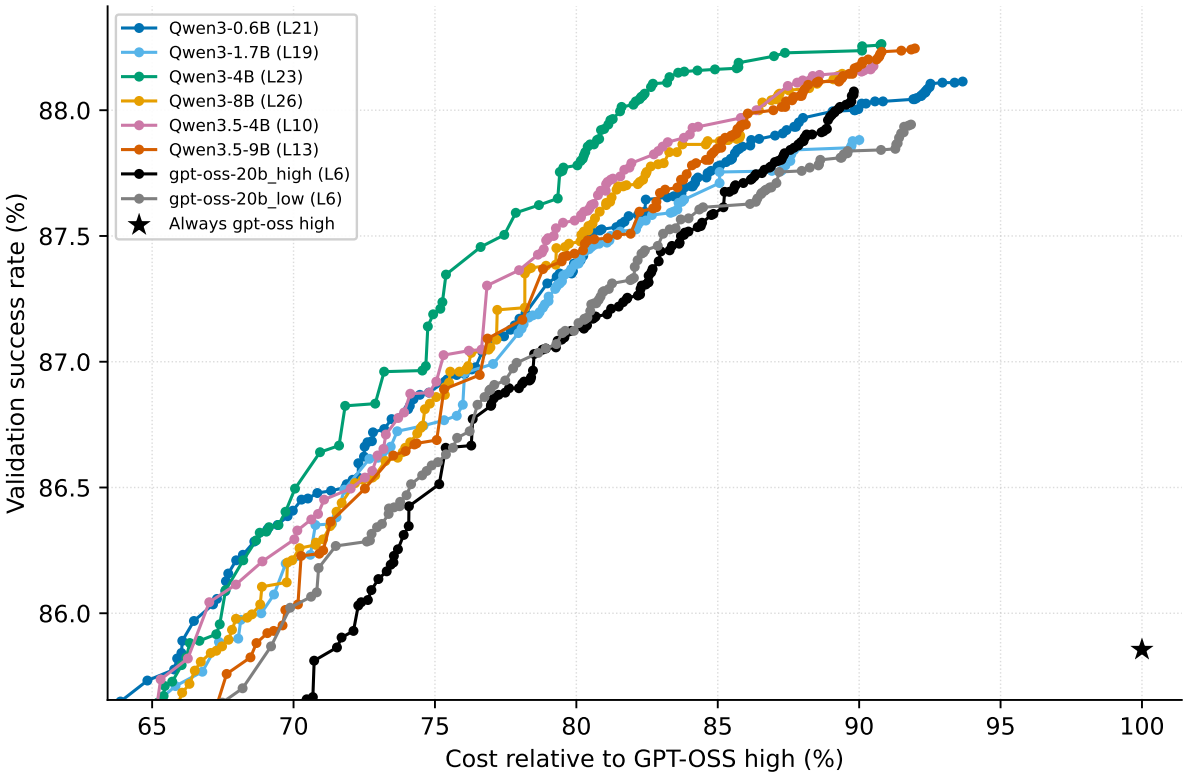


Figure B.19: Zoomed high-success region of the ETD encoder comparison. The zoom highlights the differences between encoder choices near and above the always-anchor success level.

B.4.2. Direct Routing Model Selection

Table B.2 reports the model-selection mix for direct probe routing. The pooled multilingual router is shown at both the anchor-matching and highest-success operating points. The English-only transfer router is reported only at its highest-success operating point, since it does not reach an anchor-matching point under the full cost model.

Selected model	Pooled multilingual		English-only transfer
	Matched success	Max success	Max success
Qwen3-0.6B	0.00%	0.00%	0.05%
Qwen3-1.7B	0.86%	0.73%	0.34%
Qwen3-4B	3.45%	2.32%	0.00%
Qwen3-8B	1.73%	1.23%	0.00%
Qwen3.5-4B	11.75%	3.14%	0.09%
Qwen3.5-9B	28.73%	25.82%	0.11%
gpt-oss-low	10.14%	1.59%	0.14%
gpt-oss-high	43.34%	65.18%	99.27%

Table B.2: Model-selection mix for direct probe routing on the test split. Values report the percentage of test examples assigned to each candidate model. Matched success denotes the anchor-matching operating point, while maximum success denotes the highest-success operating point. The English-only transfer router is not reported at the anchor-matching point because no such operating point is achieved.

B.4.3. ETD Routing Model Selection

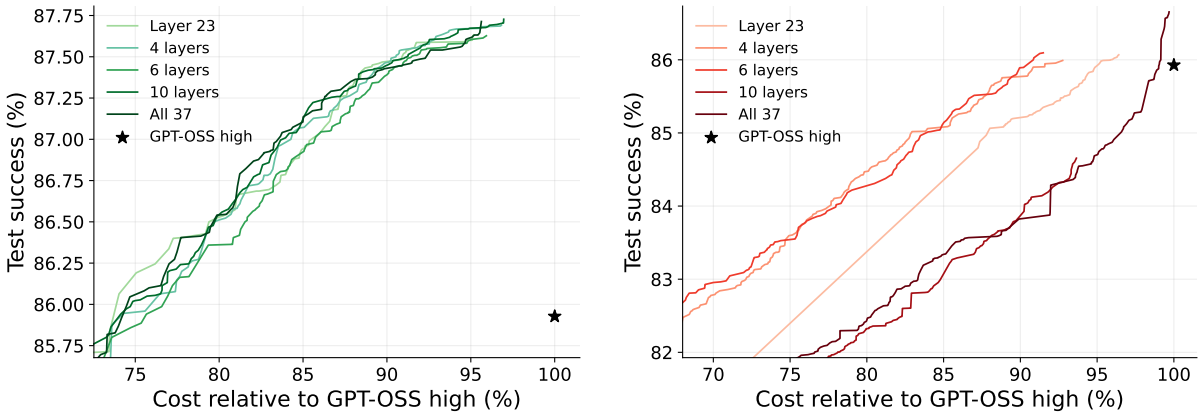
Table B.3 reports the model-selection mix for ETD routing with the Qwen3-4B encoder at layer 23. Both pooled multilingual and English-only transfer ETD are shown at the anchor-matching and highest-success operating points.

Selected model	Pooled multilingual ETD		English-only ETD	
	Matched success	Max success	Matched success	Max success
Qwen3-0.6B	0.45%	0.36%	0.09%	0.09%
Qwen3-1.7B	1.55%	0.75%	0.34%	0.27%
Qwen3-4B	3.32%	1.48%	0.27%	0.18%
Qwen3-8B	1.66%	1.25%	0.41%	0.27%
Qwen3.5-4B	19.36%	4.91%	0.27%	0.18%
Qwen3.5-9B	26.64%	12.75%	13.66%	10.39%
gpt-oss-low	18.45%	2.61%	2.48%	1.36%
gpt-oss-high	28.57%	75.89%	82.48%	87.25%

Table B.3: Model-selection mix for ETD routing on the test split. All rows use Qwen3-4B as the encoder and layer 23 as the probe layer. Values report the percentage of test examples assigned to each candidate model. Matched success denotes the anchor-matching operating point, while maximum success denotes the highest-success operating point.

B.4.4. Layer-Ensemble Routing Frontiers

Figure B.20 shows the high-success region of the layer-ensemble ETD routing frontiers. The full frontiers and representative operating points are reported in Section 4.4.3; the zoomed plots are included here to make the differences near the anchor success level easier to inspect.



(a) Pooled multilingual ETD

(b) English-only transfer ETD

Figure B.20: Zoomed high-success region of the ETD layer-ensemble routing frontiers. Each curve corresponds to a different set of `Qwen3-4B` encoder layers whose predicted success scores are averaged before routing.

B.5. Additional BFCL Results

This appendix reports the BFCL results that complement Section 4.5. Table B.4 gives the mean empirical success rate for each BFCL subset and model. The following figures report the full layer-wise AUROC trajectories for the BFCL success-prediction experiment across the Qwen-family models not shown in the main text.

BFCL subset	Qwen3-0.6B	Qwen3-1.7B	Qwen3-4B	Qwen3-8B	Qwen3.5-4B	Qwen3.5-9B
live_irrelevance	79.84	81.36	82.31	75.79	45.95	63.85
live_multiple	48.09	69.23	78.42	78.29	71.93	60.51
live_parallel	42.50	43.75	77.50	81.25	70.00	77.50
live_parallel_multiple	49.17	70.00	72.50	67.50	67.50	65.00
live_relevance	71.25	57.50	81.25	85.00	78.75	72.50
live_simple	59.69	78.68	84.26	84.19	81.40	69.38
multiple	76.80	88.10	96.20	94.70	91.30	83.50
parallel	70.00	83.60	93.00	93.30	84.20	87.40
parallel_multiple	59.20	81.20	90.40	90.40	85.10	82.60
simple_python	82.00	89.05	95.70	96.05	90.95	81.00

Table B.4: Per-subset BFCL success rates for the Qwen-family models. BFCL success is mean empirical success over five sampled generations per prompt ($k/5$), reported as a percentage. The main text reports the corresponding overall success rates on the full aligned BFCL evaluation set of 3251 examples per model across 10 BFCL subsets.

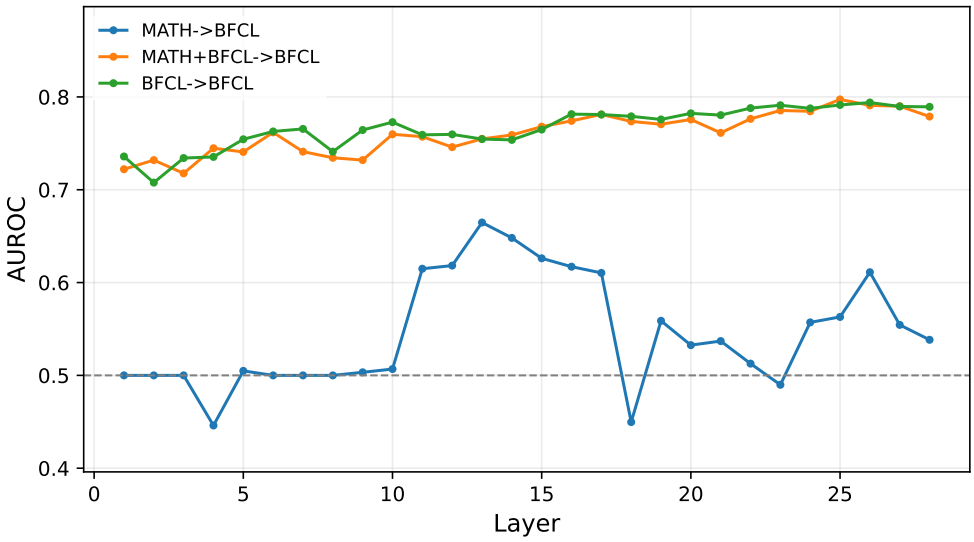
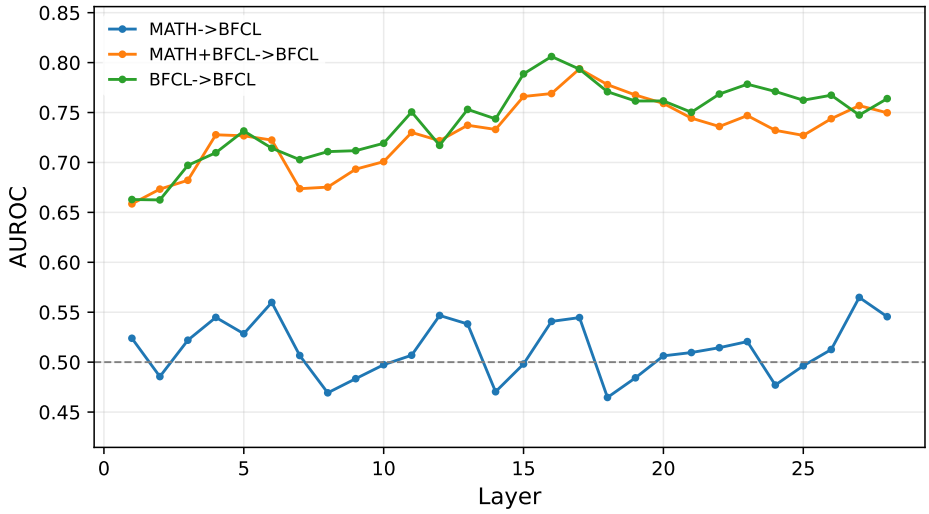
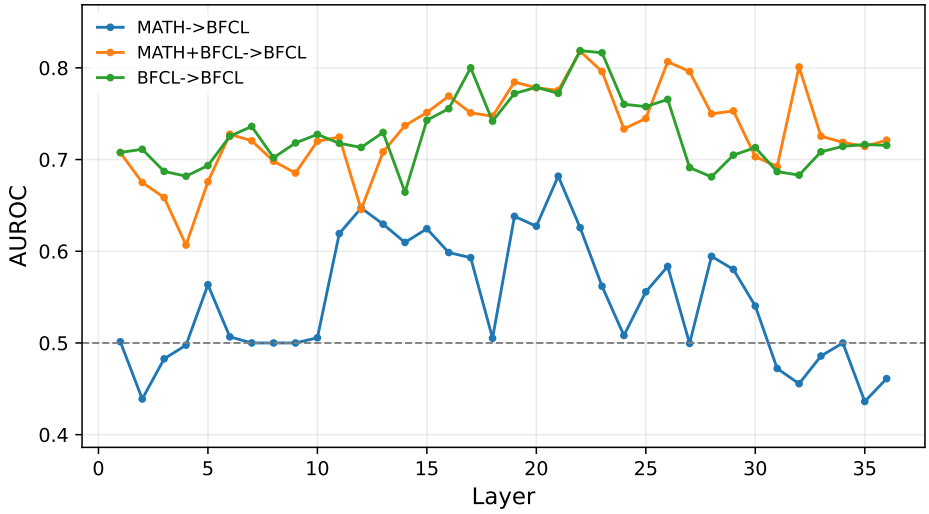


Figure B.21: Layer-wise BFCL AUROC trajectory for Qwen3-0.6B. The plot compares direct MATH-to-BFCL transfer, joint MATH+BFCL training evaluated on BFCL, and BFCL-only training evaluated on BFCL.

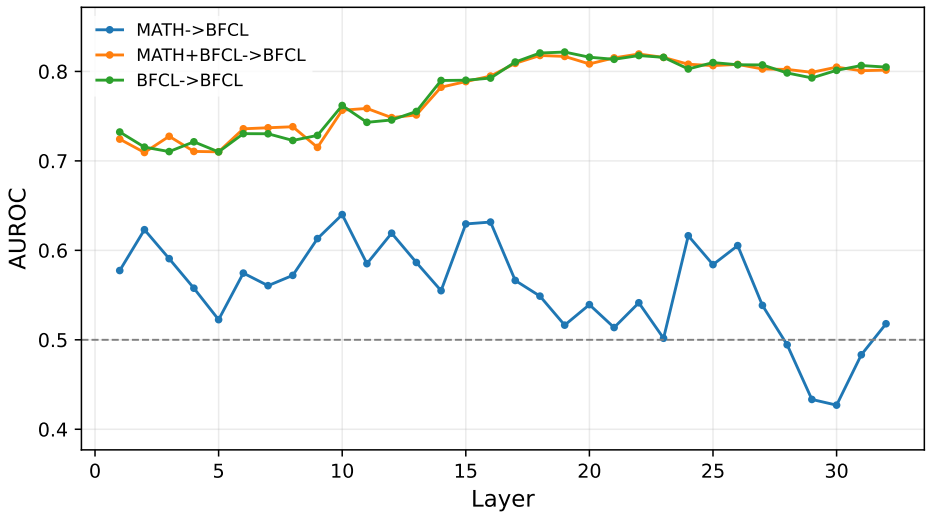


(a) `Qwen3-1.7B`

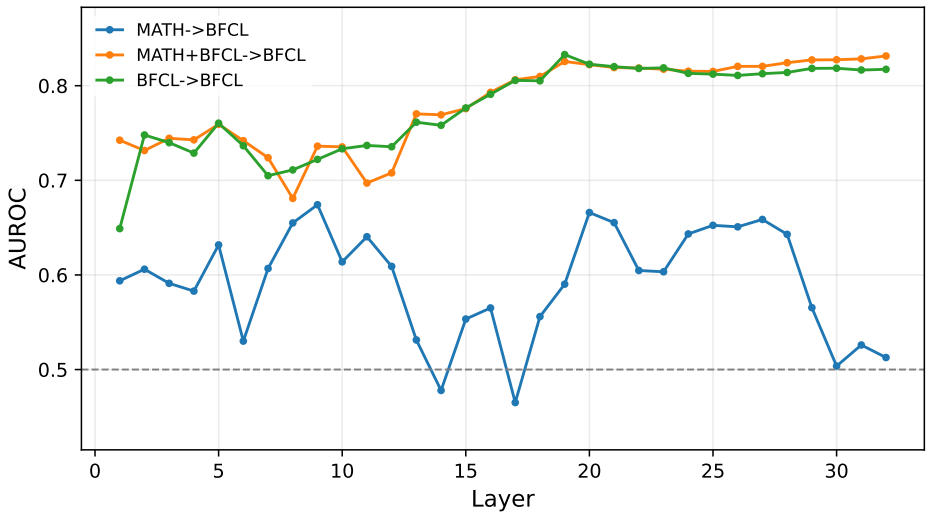


(b) `Qwen3-4B`

Figure B.22: Layer-wise BFCL AUROC trajectories for `Qwen3-1.7B` and `Qwen3-4B`. Each panel compares direct MATH-to-BFCL transfer, joint MATH+BFCL training evaluated on BFCL, and BFCL-only training evaluated on BFCL.



(a) `Qwen3.5-4B`



(b) `Qwen3.5-9B`

Figure B.23: Layer-wise BFCL AUROC trajectories for the Qwen3.5 models. Each panel compares direct MATH-to-BFCL transfer, joint MATH+BFCL training evaluated on BFCL, and BFCL-only training evaluated on BFCL. Together with Figure B.21, Figure B.22, and the main-text `Qwen3-8B` result, these plots complete the BFCL cross-dataset success-prediction results for the Qwen-family models.

List of Figures

List of Tables

Acknowledgements

I would like to express my sincere gratitude to Prof. Mark Carman and Prof. Gianluca Demartini for their guidance, feedback, and support throughout the development of this thesis. Their insights and suggestions played a fundamental role in shaping and refining this work.

I am grateful to Politecnico di Milano for giving me the opportunity to take part in the double degree programme with the University of Queensland. This experience allowed me to grow both academically and personally, and gave me the chance to carry out a part of my studies in an incredible environment.

A special thank you goes to my family and to my girlfriend, Yara, for their constant love, patience and support. Their encouragement has been fundamental throughout my studies, especially during the most challenging moments of this journey, and this achievement would not have been possible without them.

Finally, I would also like to thank the friends and colleagues who accompanied me during this journey. The discussions, encouragement, and shared moments made this path more meaningful and helped me stay motivated during the most demanding parts of it.