Agentic Discovery of Non-Canonical Antimicrobial Peptides with AMPGAN v3

Jay Hwasung Jung [1] **Xiaohan Zhang** [1] **Shenghan Song** [2] **Mahmoud Sayedahmed** [3] **Chijian Xiang** [2 4]
Yunong Xu [2] **Ahmed AbdelKhalek** [3] **Severin T. Schneebeli** [5] **Matthew J. Wargo** [6] **Jianing Li** [2] **Safwan Wshah** [1]

Abstract

Antimicrobial resistance causes to over a million deaths annually. Antimicrobial peptides (AMPs) are a promising solution, but generative AMP models are not yet ready to design peptides with non-natural amino acids and/or chemical modifications, which are essential for real-world peptide drugs. We present **AMPGAN v3**, a multi-objective conditional GAN that expands the generative vocabulary to D-amino acids and N/C-terminus modifications such as amidation. By separating adversarial and activity-aware supervision across two specialized discriminators, AMPGAN v3 substantially improves training stability and outperforms prior generative AMP models on external classifiers. We validated five candidates spanning three structural classes *in vitro*; two showed activity against Gram-positive strains, with the best candidate reaching MIC 8 μg/mL against *B. subtilis*. To support downstream curation, we further present **PepCraft**, a multi-agent framework for end-to-end AMP discovery in which a Planning Agent orchestrates specialized executors for generation, filtering, and verification. Its prioritization recommendations align with our in vitro outcomes. Together, these contributions let us examine, on a small but real scale, how generative and agentic AI compose in therapeutic peptide discovery. Code: https://github.com/marszzibros/AMPGANv3.

[1]Department of Computer Science, University of Vermont, Burlington, VT, USA [2]Borch Department of Medicinal Chemistry and Molecular Pharmacology, Purdue University, West Lafayette, IN, USA [3]Department of Comparative Pathobiology, Purdue University, West Lafayette, IN, USA [4]Department of Horticulture and Landscape Architecture, Purdue University, West Lafayette, IN, USA [5]Department of Industrial and Molecular Pharmaceutics, Purdue University, West Lafayette, IN, USA [6]Department of Microbiology and Molecular Genetics, Larner College of Medicine, University of Vermont, Burlington, VT, USA. Correspondence to: Jianing Li <jianing-li@purdue.edu>, Safwan Wshah <Safwan.Wshah@uvm.edu>.

Accepted at the 2026 Workshop on Generative and Agentic AI for Biology (ICML 2026)

1. Introduction

Despite decades of pharmaceutical investment, only five novel classes of antibiotics have been commercialized since 2000, a period often referred to as the *"discovery void"* (Silver, 2011). This slump is attributed to the high cost and low commercial returns of antibiotic development (Gargate et al., 2025). Most importantly, the emergence of antimicrobial resistance (AMR) continues to hinder the efficacy of current therapeutics (WHO, 2023), contributing to 1.27 million global deaths in 2019 (Murray et al., 2022). Addressing this escalating health crisis requires both new antimicrobial therapeutics with anti-resistance mechanisms and scalable approaches to discover them.

Antimicrobial peptides (AMPs) are a leading candidate: relatively short sequences (typically < 64 amino acids) that act through multiple mechanisms, often disrupting bacterial membranes via non-specific physical interactions,a mechanism associated with slower resistance emergence (Bucataru & Ciobanasu, 2024). AMPs are particularly suitable to data-driven *de novo* design because substantial experimentally-validated database exist (Pirtskhalava et al., 2020; Wang et al., 2026), and standard activity assays are sufficiently inexpensive to enable wet-lab validation of generated candidates.

These properties have motivated generative AI approaches for AMP discovery. Early work leveraged variational autoencoders (Szymczak et al., 2023) and adversarial architectures (Tucs et al., 2020; Ferrell et al., 2020; Van Oort et al., 2021) to produce AMP-like sequences. Subsequent studies introduced multi-objective optimization approaches (Wang et al., 2025c;b), while most recent methods adopt large language model (LLM)-based foundation models with pretraining and prompt tuning (Wang et al., 2025a). Despite this progress, two limitations persist across these approaches: (1) *In vitro* tested AMPs are predominately α-helical. Short helical peptides suffer from conformational instability and rapid proteolytic degradation, limiting their therapeutic viability (Yakimov et al., 2016). (2) Existing methods omit chemical modifications (e.g., D-amino acids, terminal modifications) from their training. In real-world therapeutic applications, these modifications are indispensable; peptides without them are highly susceptible to degradation

Table 1. Comparison of generative methods for antimicrobial peptide (AMP) discovery. AMPGAN v3 is the only method supporting D-amino acids and terminal modifications, significantly expanding the accessible chemical space beyond canonical L-amino acids.

Method	Architecture	Conditional Generation	Multi-Objective	Species Targeting	Chemical Modifications	in vitro Validation
PepGAN (Tucs et al., 2020)	GAN	✗	✗	✗	✗	✓
HydrAMP (Szymczak et al., 2023)	cVAE	✓	✗	✗	✗	✓
HMAMP (Wang et al., 2025c)	GAN (multi-D)	✓	✓	✗	✗	✓
MOFormer (Wang et al., 2025b)	Transformer	✓	✓	✗	✗	✗
AMP-Designer (Wang et al., 2025a)	LLM (GPT)	✓	✓	✓	✗	✓
AMPGAN v2 (Van Oort et al., 2021)	cGAN	✓	✗	✗	✗	✓*
AMPGAN v3 (Ours)	**cGAN (multi-D)**	✓	✓	✓	✓	✓

Note: ✓= supported, ✗= not supported, * AMPGAN v1 (Ferrell et al., 2020) includes *in vitro* validation

in minutes before showing any therapeutic effect (Lucana et al., 2021).

Among prior approaches, AMPGAN (Ferrell et al., 2020; Van Oort et al., 2021) is the work most directly related to ours: a conditional GAN that supports target-microbe conditioning. However, AMPGAN suffers from severe training instability, only $\sim$10% of random initializations produce usable models, with failed runs collapsing to repeated single-token outputs (Kossale et al., 2022). Its conditioning is also restricted to L-amino acid peptides of 32 residues or fewer, showcasing the chemical-scope limitations described above.

Improving the generative model addresses only part of the challenge. Even if a model generates highly stable sequences, translating those *in-silico* designs into experimentally testable candidates requires filtering against physicochemical constraints, and cross-referencing against known databases. These repetitive steps scale poorly with increased generative output volume when performed manually. Recent work in LLM-based scientific agents has begun to explore how such workflows can be orchestrated with minimal human intervention. While agentic systems have been applied to chemistry (Panapitiya et al., 2025) and protein engineering (Ponnapati et al., 2025; Fallahpour et al., 2026) they remain unexplored for AMP discovery. Unlike scripted pipelines, agentic systems support adaptive control flow: the Planning Agent can re-invoke executors with revised plans when intermediate results deviate from the user objective. By combining controllable generation with specialized executor agents, we explore how generative and agentic AI can be composed in an end-to-end AMP discovery pipeline.

In this work, **we present AMPGAN v3**, a multi-objective conditional generative model that jointly optimizes the sequence realism and antimicrobial activity for AMP discovery. AMPGAN v3 expands the chemical scope of prior approaches to include D-amino acids, and terminal modifications, and produces structurally diverse outputs spanning α-helical, β-hairpin, and random coil conformations, enabling the generation of peptides with chemical stability required for therapeutic viability. By separating adversarial training from activity prediction across two specialized discriminators, we resolve the training instability reported in AMPGAN v2, significantly improving successful-run rates. We validate AMPGAN v3 through *in vitro* experiments, testing five generated candidates spanning α-helical, β-hairpin, and random-coil conformations and incorporating D-amino acids and amidation, which were not previously supported by AMP generative models. Two of five candidates exhibited clear antimicrobial activity against Gram-positive species. Finally, we develop an exploratory multi-agent framework for end-to-end AMP discovery, which we refer to as **PepCraft**, demonstrating it with AMPGAN v3 as the generative component. A Planning Agent orchestrates specialized executors for sequence generation, physicochemical filtering and database verification, exploring how generative and agentic AI can be composed in scientific discovery pipelines.

2. Background and Related Work

2.1. Antimicrobial Peptides (AMPs)

AMPs are short sequences of fewer than 100 amino acids that exhibit wide ranges of activity against bacteria, fungi, and viruses by primarily targeting microbial membranes (Bucataru & Ciobanasu, 2024; Wang et al., 2026). They are generally cationic (+1 to +9) and amphipathic, facilitating electrostatic attraction to negatively charged microbial membranes and subsequent insertion into the lipid bilayer. Accordingly, AMPs commonly adopt secondary structures such as α-helices and β-sheets, which support their amphipathic architecture (Ma et al., 2024). AMPs offer several advantages over conventional antibiotics: they often act through multiple targets, reducing the likelihood of resistance development (Bucataru & Ciobanasu, 2024), and they exhibit rapid bactericidal activity with membrane disruption occurring within seconds to minutes (Fantner et al., 2010). These properties make AMPs particularly attractive candidates against multidrug-resistant (MDR) pathogens.

2.2. Generative Models for AMP Discovery

Generative models have illustrated great promise in discovering active antimicrobial peptides. Early approaches focused on balancing sequence fidelity with biological activity. PepGAN (Tucs et al., 2020) employs a generative adversarial network guided by an activity predictor to filter out inactive sequences. Similarly, HydrAMP (Szymczak et al., 2023) utilizes a conditional Variational Autoencoder (VAE) to disentangle peptide representations from antimicrobial conditions. This approach allows for both unconstrained and analogue generation. Subsequent research expanded into multi-objective optimization and large foundation models. HMAMP (Wang et al., 2025c) trains a generator against multiple discriminators to optimize distinct attributes like minimum inhibitory concentration and hemolysis. However, it relies on reward-based gradient estimation, which suffers from non-differentiable limitations. MOFormer (Wang et al., 2025b) addresses multi-objective design using a conditional Transformer with Pareto-based feedback. Taking a different route, AMP-Designer (Wang et al., 2025a) adapts a large language model architecture. It pretrains on vast peptide databases to achieve high *in vitro* and *in vivo* efficacy. For controllable design, AMPGAN v2 (Van Oort et al., 2021) adopted bidirectional conditional GANs. It conditions peptide generation on specific target microbes and mechanisms. The generated peptides were highly diverse and yielded a high predicted-active rate. A comprehensive comparison of these methods is provided in Table 1.

2.3. Agentic Systems in Computational Biology

Large Language Models (LLMs) are now driving autonomous agentic systems. These systems orchestrate tools, reason over data, and refine outputs. As a result, they are reshaping computational biology and wet-lab workflows. Recent works such as AutoLabs (Panapitiya et al., 2025) break down complex experiments into tasks for specialized, self-correcting agents. It translates natural language into hardware-ready protocols for liquid-handling robots. Similarly, ProteinCrow (Ponnapati et al., 2025) applies this approach directly to protein engineering. It equips an LLM with curated tools for structure prediction and inverse design, allowing autonomous execution that previously requires extensive manual oversight. BioReason-pro (Fallahpour et al., 2026) demonstrates that LLMs equipped with bioinformatics tools can produce natural-language interpretations of biological sequences.

3. Materials and Methods

3.1. *In Vitro* Protocol

To validate the biological activity of the generated antimicrobial peptide candidates, selected peptides were synthesized and tested using standard MIC and MBC assays. Peptides were dissolved in sterile water to prepare 10 mg/mL stock solutions. Bacterial strains used for evaluation are summarized in Appendix A. Strains were cultured overnight at 37 °C under aerobic conditions using tryptic soy medium, except for Enterococcus faecium, which was cultured using brain heart infusion medium. Bacterial cultures were diluted to approximately 5×10^5CFU/mL before peptide treatment.

MIC values were determined using a two-fold serial dilution assay in 96-well plates. Peptide dilutions were mixed with bacterial suspensions and incubated at 37 °C for 16 h. The MIC was defined as the lowest peptide concentration that resulted in no visible bacterial growth. To determine MBC values, samples from wells without visible growth were plated onto antibiotic-free agar and incubated under the same conditions. The MBC was defined as the lowest peptide concentration that prevented bacterial regrowth. MIC and MBC results are summarized in Table 2.

3.2. AMPGAN-v3 Model Architecture

We formulate conditional AMP generation as a multi-objective adversarial learning task. Given a training set $\mathcal{D} = \{x_i, c_i\}_{i=1}^{N}$, where $x_i \in \{0,1\}^{L \times V}$ is a one-hot encoded peptide sequence of maximum length L over a vocabulary of V tokens, and $c_i = (s_i, m_i, \ell_i)$ is a condition tuple comprising target species s_i, minimum inhibitory concentration (MIC) m_i, and peptide length ℓ_i. We train a generator G against two specific discriminators: D_{adv} for adversarial training and D_{mic} for antimicrobial activity regression. The overall architecture of AMPGANv3 is shown in Figure 1.

Generator: The generator $G_\theta : \mathcal{Z} \times \mathcal{C} \to \mathbb{R}^{L \times V}$ maps a latent vector $z \sim \mathcal{N}(0, I_{d_z})$ and condition $c = (s, m, \ell)$ to a sequence of token logits. Each condition is embedded into a shared space $\mathbb{R}^{d_e}$: species s via a learned embedding table, and the MIC value m and normalized length ℓ via learned linear projections. The combined condition vector is their concatenation $\mathbf{c} = [\mathbf{e}_s; \mathbf{e}_m; \mathbf{e}_\ell] \in \mathbb{R}^{3d_e}$. We modulate the latent code via Feature-wise Linear Modulation (FiLM) (Perez et al., 2018) with gating: $z' = \sigma(\alpha) \cdot z + \gamma(c) \odot z + \beta(c)$ where γ, β are learned affine transformations of c and α is a learnable gating scalar. The modulated vector is progressively upsampled through transposed convolutions to length L, then refined by a Transformer encoder. A final convolutional stack projects to logit space $\mathbb{R}^{L \times V}$.

Discriminators: We separate adversarial and activity-aware supervision into two discriminators rather than coupling them into a single conditional Discriminator. First, the two signals answer different questions: D_{adv} asks whether a sequence is plausibly peptide-like, a distributional judgment over the full training corpus, whereas D_{mic} asks whether a sequence is consistent with a specified potency, a conditional judgment that depends on the target MIC. Both discrimi-

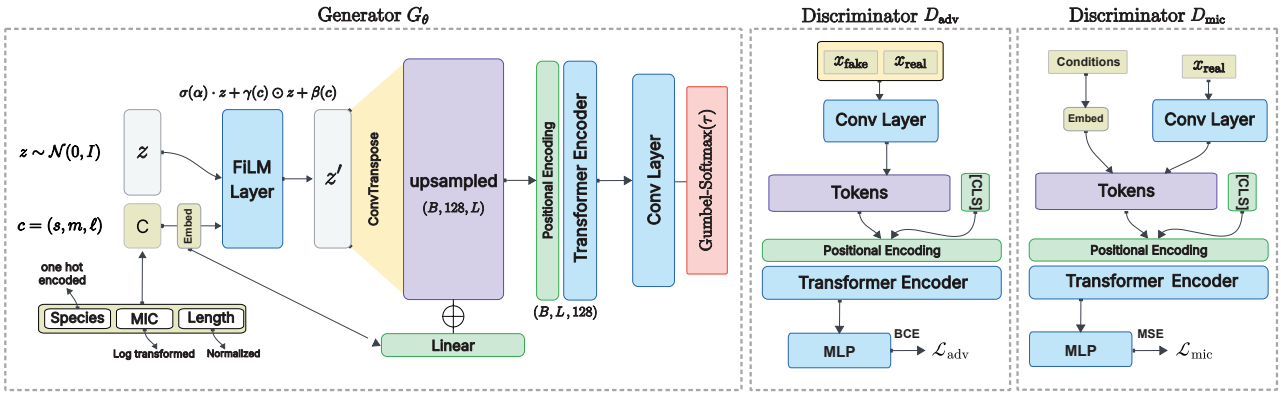

Figure 1. **AMPGAN-v3 architecture.** The generator G_θ maps a latent vector $z \sim \mathcal{N}(0, I)$ and condition $c = (\text{species} - s, \text{MIC} - m, \text{length} - \ell)$ to a sequence of token logits via FiLM-based conditioning, transposed convolutions, and a Transformer encoder. Two discriminators are trained jointly: D_{adv} classifies real vs. generated sequences unconditionally, whereas D_{mic} regresses MIC values from real sequences conditioned on species.

nators share a convolutional stem that projects the one-hot token dimension into a subspace with dimension d, followed by a Transformer encoder with a learnable [CLS] token prepended to the sequence. Then, the [CLS] representation is passed through an MLP head for prediction. D_{adv} operates unconditionally on both real and generated sequences and outputs a binary real/fake classification. D_{mic} is trained exclusively on real sequences, conditions on species by an extra embedding token to the sequence before the Transformer encoder, and regresses a MIC value. Its parameters are updated exclusively on real sequences with experimentally measured MIC labels; during generator training,D_{mic} is frozen and scores generated sequences to provide gradients to Generator.

Multi-Objective Training: The generator is trained against both discriminators simultaneously. At each training step, we first update D_{adv} on real and generated samples, then update D_{mic} on real samples only, and finally update G on a weighted combination of four objectives: $\mathcal{L}_G = \lambda_\ell \mathcal{L}_{\text{len}} + \lambda_s \mathcal{L}_{\text{sim}} + \lambda_a \mathcal{L}_{\text{adv}} + \lambda_m \mathcal{L}_{\text{mic}}$ where $\mathcal{L}_{\text{adv}}$ and $\mathcal{L}_{\text{mic}}$ are computed with discriminator parameters frozen. Since G produces continuous logits over a discrete vocabulary, we apply the Gumbel-Softmax estimator with temperature τ to obtain differentiable discrete samples $\hat{x} = \text{Gumbel-Softmax}_\tau(G(z, c))$ for both discriminators. The adversarial term is the standard cross-entropy (CE) against D_{adv}, and the conditional term is mean-squared error (MSE) that pushes generated sequences toward the target potency c as scored by D_{mic} conditioned on species s:

$$\begin{aligned} \mathcal{L}_{\text{adv}} &= -\frac{1}{B} \sum_i \log D_{\text{adv}}(\hat{x}_i), \\ \mathcal{L}_{\text{mic}} &= \frac{1}{B} \sum_i \| D_{\text{mic}}(\hat{x}_i, s_i) - c_i \|^2. \end{aligned} \tag{1}$$

The remaining two terms operate directly on the generator's predicted token distribution $\hat{p}_{i,t,v}$: the probability assigned to vocabulary token v at position t in sequence i. For each sample i, let y_i be the real sequence with tokens $y_{i,t}$ at position t and EOS at position ℓ_i, and full sequence length $L_i = \ell_i + 1$ accounting for a special terminus token (See Section 4 for details). The length loss signals sequence termination by maximizing the predicted EOS probability at each sequence's true EOS position:

$$\mathcal{L}_{\text{len}} = -\frac{1}{B} \sum_i \log \hat{p}_{i,\ell_i,\text{EOS}} \tag{2}$$

The similarity loss $\mathcal{L}_{\text{sim}}$ extends this per-position supervision across the full sequence, providing a token-level reconstruction signal that stabilizes the early training. Sequences positions are weighted at 1 and padding positions at $\alpha = 0.1$, so the peptide reconstruction signal dominates while trailing tokens are mildly discouraged.

$$\mathcal{L}_{\text{sim}} = -\frac{\sum_{i,t} w_{i,t} \log \hat{p}_{i,t,y_{i,t}}}{\sum_{i,t} w_{i,t}}, \quad w_{i,t} = \begin{cases} 1 & t \le L_i \\ \alpha & t > L_i \end{cases} \tag{3}$$

3.3. Agentic Discovery Pipeline

We propose a multi-agent framework for automated AMP design, orchestrated by a Planning Agent $\mathcal{P}$ that dynamically selects and instructs specialized executor agents based on iterative feedback as illustrated in Figure 2. Specifically, at each iteration t, the Planning Agent receives the user objective θ_0 and the previous report r_{t-1} where $r_0 = \emptyset$. It then selects an executor agent e and prompts it with a specific instruction i_t^e at iteration t.

$$\mathcal{P}(\theta_0,\ r_{t-1}) = \begin{cases} \text{END} & \text{If Objective Fulfilled} \\ (e, i_t^e) & \text{Otherwise} \end{cases}. \tag{4}$$

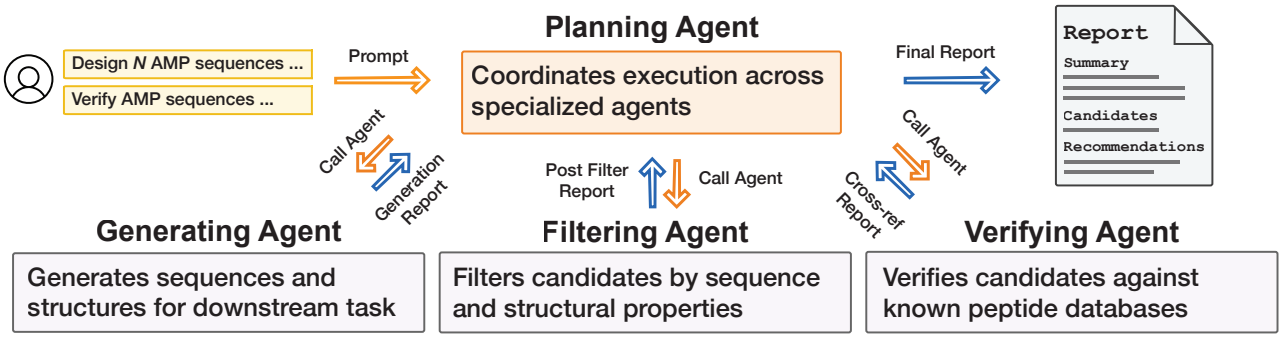

Figure 2. **Agentic workflow for AMP discovery.** A Planning Agent translates user objectives into natural-language instructions and send them to specialized executors with tools: Generating Agent, Filtering Agent and Verifying Agent. Each executor returns a report, and Planning Agent iterates until the objective is met.

The selected executor e executes its designated tools along with the instruction prompt generated by the planning agent $\mathcal{P}$ and returns a run report $r_t = e(i_t^e)$. Example instructions and reports are provided in Appendix C.

Planning Agent: Decomposes the user-defined objective into natural language instructions and calls the appropriate executor. Upon receiving a run report r_t, it decides whether to re-invoke an executor with revised parameters or proceed to the next stage.

Generating Agent. Translates user-defined natural-language objectives into structured generation calls for AMPGAN v3 and SimpleFold (Wang et al., 2025d)-based structure prediction.

Filtering Agent. Applies sequential physicochemical and structural screens to refine the raw candidate pool. Property-level filters (charge, hydrophobicity, length) run first, and structure-dependent filters run only on candidates that survive earlier passes. Each filtering cycle concludes with a status check that the Planning Agent uses to plan.

Verifying Agent. Assesses novelty and biological relevance through BLAST (Camacho et al., 2009) homology search against SwissProt (Boutet et al., 2007) (broad protein context) and the AMPGAN v3 training dataset (DBAASP (Pirtskhalava et al., 2020); near-duplicate detection and known-activity recovery). Annotated hits are returned to the Planner as the basis for prioritization recommendations.

Agent system prompts, tool specifications, and a representative trajectory are provided in Appendix C.

4. Dataset Curation

To enable the training of our AMPGAN-v3, we curate a dataset of experimentally validated AMP sequences collected from DBAASP v3 (Pirtskhalava et al., 2020) via its public REST API. We queried all peptide IDs sequentially, retaining only monomers with sequences of 64 residues or fewer. For each retained entry, we extracted the amino acid sequence, N-terminus and C-terminus modifications, and per-species antimicrobial activity, retaining only MIC and IC measurements. Concentration values reported as ranges were replaced by their midpoint, and all concentrations were converted to μg/mL. Entries with missing units or unparseable concentration values were excluded. Sequences were stored in their original case to preserve D-amino acid annotations, and the resulting dataset was deduplicated by sequence. Each sequence is encoded as a structured token sequence of the form [N-term][SOS] [Amino Acid Tokens] [EOS][C-term], where [N-term] and [C-term] denote terminus-modification tokens (or none) and [SOS]/[EOS] mark sequence boundaries. This format allows the model to generate terminus modifications and sequence content jointly within a single output stream.

The final training dataset contains 34,275 rows corresponding to 10,892 unique sequences. A single sequence may appear multiple times, each paired with a distinct combination of N-terminus modification, C-terminus modification, or target species. The six target species are *E. Coli*, *P. Aeruginosa*, *K. Pneumoniae*, *S. Aureus*, *B. Subtilis*, and *S. Epidermidis*. During training, each row is treated as an independent sample, allowing the model to learn the mapping from a sequence-species pair to its inhibitory activity (MIC).

In contrast to AMPGAN v2 (Van Oort et al., 2021), which assigned arbitrarily high MIC values to represent inactivity, our approach relies entirely on experimentally measured concentrations. Antimicrobial activity is inherently species-dependent. Because a sequence can be highly effective against one species while remaining inactive against another, presetting MIC values for untested pairs would inject misleading supervision signals into the training process.

5. Experiments

We evaluate AMPGAN v3 along three stages: (1) *in vitro* antimicrobial activity of synthesized candidates against clin-

ically relevant bacterial strains, (2) *in-silico* generation quality, comparing against AMPGAN v2 (Van Oort et al., 2021) and published sequences from the reviewed methods Section 2.2, and (3) the utility of our agentic workflow relative to single-prompt baselines. More details can be found in Appendix B.

5.1. *In Vitro* Evaluation

Candidate selection. From AMPGAN v3 generations conditioned on *E.Coli* and *S.Aureus* with target MIC $<$ 32 μg/mL, we filtered candidates using four physicochemical criteria common in AMP screening: instability index $<$ 50, net charge at pH 7 between 2 and 9, GRAVY score between -1 and 1, and isoelectric point $>$ 9. Secondary structure was predicted with AlphaFold3 (Abramson et al., 2024), and chemical modifications (D-amino acid substitutions, C-terminal amidation) were introduced using Schrödinger Maestro (Schrödinger, LLC, 2025). From the filtered pool, we selected five candidates spanning three structural classes: two α-helices (HY-P60322, HY-P60323), two β -hairpins (HY-P60324, HY-P60325), and one random coil (HY-P60326). The selected candidates and their target species are listed in Table 2.

Results. Five AMPGAN v3 candidates were synthesized and evaluated against clinically relevant bacterial strains. Minimum inhibitory and bactericidal concentrations (MIC/MBC) are reported in Table 2, alongside thanatin and conventional antibiotic controls. Two of five candidates exhibited antimicrobial activity. HY-P60322 showed potent activity against the Gram-positive strains *B. subtilis* (MIC/MBC = 8/8 μg/mL) and *S. epidermidis* (16/32 μg/mL). HY-P60325 showed activity against *B. subtilis* (16/32 μg/mL) and *S. epidermidis* (64/64 μg/mL). The remaining three candidates (HY-P60323, HY-P60324, HY-P60326) showed no activity ($>$ 128 μg/mL).

These results confirm that AMPGAN v3 generates sequences with genuine antimicrobial function, including across non-canonical chemistry (D-amino acids, C-terminal amidation) not supported by prior generative AMP models. However, neither active candidate inhibited growth of the species used as its conditioning target: HY-P60322 (*E. coli*-conditioned) was inactive against *E. coli*, and HY-P60325 (*S. aureus*-conditioned) was inactive against *S. aureus*. Instead, both targeted Gram-positive species absent from their conditioning. Target-species conditioning therefore did not translate to species-selective activity.

5.2. *In Silico* Evaluation

Generated Samples Quality Analysis: Figure 3 compares the generated and training AMP distributions across three views: per-residue relative frequencies (top), net charge at pH 7 (center), and GRAVY hydrophobicity (bottom).

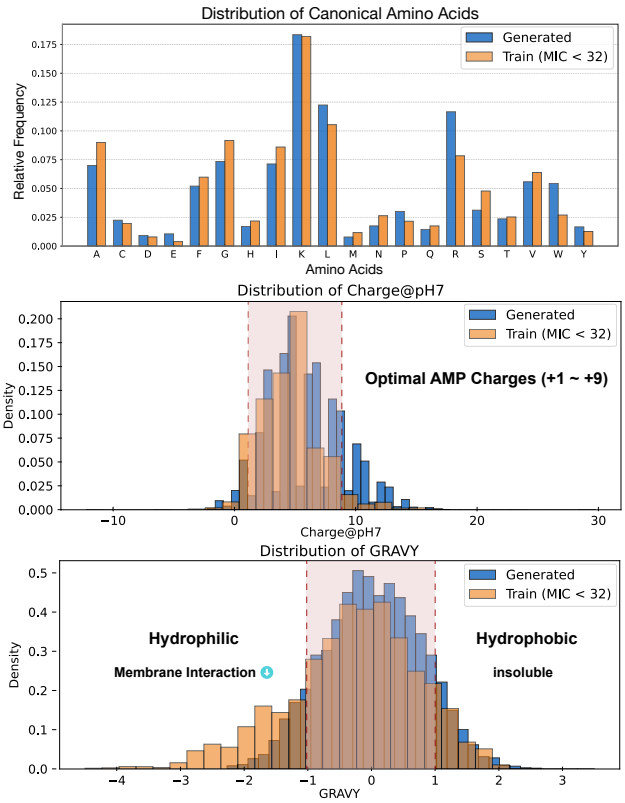

Figure 3. **Sequence property distributions for generated and training AMPs.** Per-residue frequencies (Top), net charge at pH 7 (center), and GRAVY hydrophobicity (Bottom), Generated sequences concentrate within AMP-active ranges charge in [+1, +9]; GRAVY in [-1, +1]).*Top:* per-residue relative frequencies closely match the training distribution, except for Arg (R) and Trp (W), which are overrepresented by $\sim$ 0.025.

AMPGAN v3 closely matches the training distribution on all three axes. At the residue level, the only notable deviations are Arginine (R) and Tryptophan (W), each overrepresented by approximately 0.025 (2.5 percentage points). At the sequence level, generated samples concentrate within the ranges associated with AMP activity, net charge in [+1,+9] and GRAVY in [-1,+1], preserving the cationic, amphipathic character required for selective interaction with negatively charged bacterial membranes.

Training Stability: GAN training is known to suffer from mode collapse and instability (Kossale et al., 2022). We measure training stability across 30 random seeds for both AMPGAN v2 and AMPGAN v3 under matched experimental conditions, following the criterion of realistic sequence entropy and length-controllable generation defined in (Van Oort et al., 2021). AMPGAN v2 succeeds in only 3/30 runs, consistent with the $\sim$10% rate reported in the original work. Failed AMPGAN v2 runs typically collapse to repeating a single token across the entire output. AMPGAN v3 introduces structured tokens, which shift the dominant

Table 2. Minimum inhibitory and bactericidal concentrations (MIC/MBC, µg/mL) of peptides against the bacterial panel.

Sequence code	Full sequences	*Staphylococcus aureus* USA300-0114 Wound[a]	*Bacillus subtilis* — —	*Staphylococcus epidermidis* ATCC 35984 Catheter	*Enterococcus faecium* HM-952 Human, USA
HY-P60323	KRTASKVKKHkRNAATIAVAhAR-NH_2	>128	>128	>128	>128
HY-P60324	IGCRCIQAGaGRLFVERr-NH_2	>128	>128	>128	>128
HY-P60325	SkCRVLFFPFIFVRPHGSMFCCR-NH_2	>128	**16/32**	**64/64**	>128
HY-P60322	KKRKHLRKAAGWayKKVGLWAL-NH_2	>128	**8/8**	**16/32**	>128
HY-P60326	WFPHIARAKNGVRYAkNKKGKTVLK-NH_2	>128	>128	>128	>128
Thanatin	GSKKPVPIIYCNRRTGKCQRM-NH_2	>128	>128	>128	>128

Table 3. Training stability across $N = 30$ random seeds. Both criteria measure post-training output quality: realistic, length-controllable generation. AMPGAN v2 numbers are reported in (Van Oort et al., 2021).

Model	Criterion	Successful runs
AMPGAN v2	entropy + length control	3/30
AMPGAN v3	$p \geq 70\%$	21/30
	$p \geq 80\%$	19/30
	$p \geq 90\%$	13/30

failure mode from global collapse to local formatting errors, such as incorrect placement of SOS, EOS, or terminus tokens. To quantify stability, we evaluate the proportion p of generated sequences per run that satisfy the output validity criterion (i.e., containing a valid amino acid span with length error within 3 residues and properly structured tokens).

As shown in Table 3, AMPGAN v3 achieves substantially higher stability across all thresholds. At $p \geq 70\%$, 21/30 runs are successful. Performance remains robust at stricter thresholds, with 19/30 runs at $p \geq 80\%$ and 13/30 runs at $p \geq 90\%$ (see Table 3 for AMPGAN v2 comparisons). These results indicate that AMPGAN v3 not only mitigates catastrophic mode collapse but also produces consistently valid outputs across the majority of training runs.

Table 4. Predicted AMP rates (%) on $N = 2{,}000$ unconditional generations from each model.

Model	AMP Scanner v2	amPEP
AMPGAN v3	**92.7**	**95.1**
AMPGAN v2	80.0	83.4
HydrAMP	81.0	83.0
AMP-Designer	92.4	83.9

External Predictor Scores: To assess whether AMPGAN v3 generates AMP-like sequences, we classified generated outputs using two independent predictors that differ in both architecture and training data: AMP Scanner v2 (Veltri et al., 2018), a Conv+LSTM deep learning classifier, and amPEP (Bhadra et al., 2018), a random forest model based on amino-acid distribution patterns. For comparison, we generated sequences from AMPGAN v2 (Van Oort et al., 2021) and HydrAMP (Szymczak et al., 2023) under their default settings, and utilized the pre-generated sequences released with AMP-Designer (Wang et al., 2025a). For each model, we report the fraction of $N = 2{,}000$ unconditional generations classified as AMP (Table 4).

AMPGAN v3 achieves the highest predicted-AMP rate on both classifiers (92.7% / 95.1%), outperforming AMPGAN v2 (80.0% / 83.4%), HydrAMP (81.0% / 83.0%), and AMP-Designer (92.4% / 83.9%). Agreement across two architecturally distinct predictors suggests the gain is not an artifact of either classifier's inductive bias. We hypothesize that the multi-discriminator design contributes to this improvement by separating adversarial from activity-aware supervision.

Embedding-space Distribution Analysis: Beyond per-residue and physicochemical statistics, we evaluate whether AMPGAN v3 generations occupy the same region of sequence space as real AMPs. We embed real AMPs, random sequences, and generations from AMPGAN v2 (Van Oort et al., 2021), HydrAMP (Szymczak et al., 2023), AMP-Designer (Wang et al., 2025a), and AMPGAN v3 using ESM2 (Lin et al., 2023). These embeddings are then projected to 32 PCA components, and estimate Kullback-Leibler divergences (k=3) against the real distribution using a k-nearest neighbor density ratio estimator (Zhang, 2016). Forward $KL(\text{real}|\text{gen})$ measures coverage (i.e., How well generations span the real distribution) while reverse $KL(\text{gen}|\text{real})$ measures fidelity (i.e., Whether generations remain on the real-AMP manifold).

All four generative models substantially outperform random sequences (symmetric KL 4.99-11.88 vs. 16.64). Compared to its predecessor, V3 significantly improves fidelity (reverse KL 6.43 vs 7.56, paired bootstrap 95% CI on the difference $[+0.72, +1.56]$) while matching V2's coverage (forward KL difference 95% CI $[-0.42, +0.75]$, not significant). V3 also outperforms HydrAMP on both axes (forward KL difference $[+3.14, +4.90]$, reverse KL $[+1.82, +2.91]$, both 95% CIs excluding zero). AMP-Designer achieves the lowest divergence on both axes (reverse KL 2.75; paired difference vs v3 $[-4.01, -3.20]$), which we attribute to its

Table 5. KL divergence between generations and real AMPs in ESM2 embedding space (k=10, 32-d PCA, n=2000/group). Lower is better. Δ values are paired bootstrap differences (baseline − v3) with 95% CI over 1000 resamples; positive favors v3.

	Forward ↓ (coverage)	Reverse ↓ (fidelity)
v3 (Ours)	10.90	6.43
v2	11.08	7.56
HydrAMP	14.94	8.82
AMP-Designer	**7.22**	**2.75**
Random	22.08	11.20
Δ (v2−v3)	+0.18	**+1.13**
95% CI	[−0.42, +0.75]	[**+0.72**, **+1.56**]
Δ (HydrAMP−v3)	**+4.04**	**+2.39**
95% CI	[**+3.14**, **+4.90**]	[**+1.82**, **+2.91**]
Δ (AMP-Designer−v3)	−3.68	−3.68
95% CI	[−4.14, −3.22]	[−4.01, −3.20]

initialization from AMP-GPT, a transformer pretrained on 600K+ UniProt peptides, approximately 63× the size of our labeled training set, providing a strong protein prior that aligns closely with ESM2's pretraining distribution. AMPGAN v3, trained from scratch on ∼10K AMPs without protein prior pretraining, is structurally disadvantaged on this metric independent of antimicrobial properties.

5.3. Agentic Workflow Evaluation

We present preliminary observations on the agentic workflow, comparing our multi-agent system against two single-agent baselines that progressively combine planning and execution into one model, allowing us to examine which architectural choices affect efficiency. Our goal is to characterize how the pipeline behaves on a representative AMP discovery task and to illustrate the kind of output it produces, leaving prospective evaluation of agent-prioritized candidates to future work.

Settings: (1) Single-agent (batched): One LLM plans and executes inline, issuing multiple tool calls per iteration. Tests whether unified planning and execution in a single model is more efficient than decomposition. (2) Single-agent (stepwise): Same single LLM, but restricted to exactly one tool call per iteration. Isolates the effect of role decomposition by removing batched execution while retaining a unified model. **(3) Multi-agent (Ours):** Planning and execution decomposed across separate models, with role-specific tool subsets per executor.

All settings receive the same objective: generate K candidate AMP sequences with D-amino acids targeting *E.coli* with α-helical secondary structure. Single-agent baselines use *Gemini 3.1 Pro Preview* (Google Cloud, 2026b) throughout, since planning is required at every step. The multi-agent setting uses *Gemini 3.1 Pro Preview* for the Planner and *Gemini 3.1 Flash Lite Preview* (Google Cloud, 2026a) for executors. All runs across all settings returned the requested K candidates, so cost metrics are directly comparable. We exclude `Filter_Status` calls from tool-call counts: it is a bookkeeping tool the agent invokes after each filtering request to track progress, but single-agent settings invoke it redundantly after each filter call, inflating their counts. Removing these calls ensures the comparison reflects meaningful tool calls rather than bookkeeping overhead.

Tool Call Efficiency: We ran each setting across $N = 10$ independent seeds at three task sizes $K = \{5, 10, 20\}$ requested candidates. Each tool call corresponds to one decision-execution cycle: The LLM selects a tool, the tool executes, and the result is returned to the agent's context for the next decision (see Section 3.3 for more details). Figure 5 (Top) shows total tool calls per run. Multi-agent uses fewer tool calls than batch across all task sizes. Both batch and stepwise grow and show substantially wider bootstrap intervals as K increases. Decomposing $K = 20$ by agent role (Figure 5, Bottom) localizes the multi-agent advantage to the verification stage; filtering and generation are comparable across all three settings.

Table 6. Output token usage per run across orchestration settings ($K = 20, N = 10$ seeds). Multi-agent splits generation between a Planner (large model) and Executors (small model), enabling cost-effective routing. Single-agent settings must route all tokens through a large model.

Setting	**Mean**	**Median**	**Std**
Multi-agent (Ours)			
Planner (large)	3,724	3,226	1,298
Executors (small)	12,585	11,044	3,037
Single-agent (large model only)			
Batched	9,942	9,400	3,579
Stepwise	24,845	21,169	13,733

Cost-effective model routing. Tool-call counts alone understate the cost difference between settings. Table 6 reports output token usage per run at $K = 20$. In the multi-agent setting, the Planner produces only ∼3.7K tokens per run while Executors produce ∼12.6K. However, Executors merely structure tool calls following the Planner's instructions, so they can be backed by a smaller, cheaper model. Only the Planner, which performs cross-step reasoning, requires a large model. The single-agent baselines have no such separation: batched produces ∼9.9K and stepwise ∼24.8K tokens, all of which must route through the large model. This model flexibility, large model for planning and small model for execution, is the architectural advantage that distinguishes multi-agent from single-agent in practice.

Qualitative Results: Beyond orchestration cost, we examine the content of the multi-agent workflow's output. Figure 4 shows a representative final report generated by

Summary

A total of 5 generated candidate sequences were evaluated against the SwissProt and DBAASP databases ... the candidate set exhibits varying potential; one sequence aligns with a known, highly potent antimicrobial peptide ...

Candidate-by-Candidate Findings

`C1: KKRKHLRKAAGWayKKVGLWAL` → **BLAST Match / Homology / Activity**

This sequence yielded a notable hit in the DBAASP database to the peptide *PGG* ... demonstrates strong broad-spectrum efficacy, with reported minimum inhibitory concentrations (MIC) of 1 µM against *Salmonella enterica subsp. enterica serovar Typhimurium* and 2 µM against *Escherichia coli* ...When cross-referenced with SwissProt, the sequence showed weak homology (43.33% identity, E-value 2.45) to the bifunctional purine biosynthesis protein PurH from the bacterium *Acidithiobacillus*...

Aggregate Observations

Across the set of 5 generated candidates ...only 1 of those matches (*PGG* corresponding to Candidate 1) exhibited potent antimicrobial activity at therapeutically relevant concentrations (1-2 µM). Furthermore, none of the SwissProt hits aligned with naturally occurring immune or host defense peptides; instead, they all mapped to metabolic or structural proteins across diverse taxa ...

Caveats and Recommendations

Recommendations for Next Steps: Candidate 1 is the only sequence that warrants immediate wet-lab prioritization based on the provided cross-referencing. For Candidates 3 and 5 ... it is highly recommended to perform 3D structural prediction ... Candidates 2 and 4 should be deprioritized.

Figure 4. **Multi-agent final report.** The Planning Agent synthesizes BLAST hits from the Verifying Agent into per-candidate findings, aggregate observations, and prioritization recommendations. Representative trajectory can be found in Appendix C.5

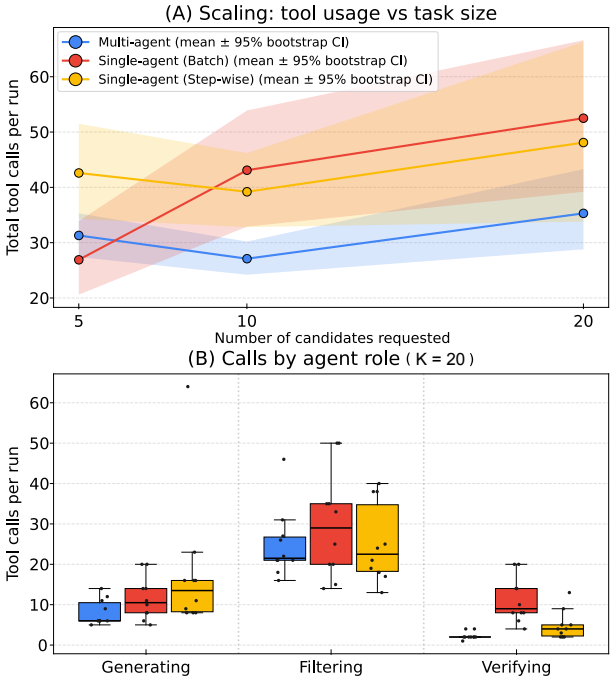

Figure 5. **Total tool calls per run as a function of task size.** Each setting was run across $N = 10$ independent seeds at $K \in \{5, 10, 20\}$ requested candidates. Lines show per-seed mean tool calls; shaded bands show 95% bootstrap confidence intervals; individual points show per-seed counts.

running the Verification pipeline on the five *in vitro* candidates from Section 5.1. The Verifying Agent retrieves BLAST (Camacho et al., 2009) hits from SwissProt (Boutet et al., 2007) and DBAASP (Pirtskhalava et al., 2020) for each candidate, and the Planning Agent synthesizes these into per-candidate findings, aggregate observations, and prioritization recommendations. For HY-P60322, the report highlights a DBAASP hit to PGG (reported MIC 1–2 μM against S. enterica and E. coli) and recommends wet-lab prioritization. HY-P60325 is deprioritized after mapping to a sequence with no reported antimicrobial activity. The report's prioritization is consistent with our *in vitro* results: HY-P60322 was one of the two active candidates, while three of the four candidates flagged for deprioritization or further analysis showed no activity. This result qualitatively supports the effectiveness of PepCraft. Prospective evaluation of selection ability remains a task for future research.

6. Conclusion and Future Work

In this study, we propose AMPGAN v3, which generates antimicrobial peptides spanning canonical and non-canonical chemistry, including D-amino acids and N/C-terminus modifications. Beyond expanding the chemical space, AMPGAN v3 improves training stability from 10% to 70% (Table 3) of its predecessor and outperforms competing models in producing AMP-like sequences (Table 4). Wet-lab evaluation of five candidates yields two actives, notably HY-P60322 with an MIC of 8 μg/mL against *B. subtilis* (Table 2), though neither active candidate inhibited its conditioning target species. While our proposed PepCraft framework is preliminary, the qualitative consistency between the agent's prioritization and these *in vitro* outcomes is encouraging (Figure 4). We anticipate PepCraft will grow as the ecosystem expands to include automated model training, molecular dynamics simulations, and broader peptide design tasks.

Future work will prioritize wet-lab validation of agent-selected candidates and the refinement of target-aware peptide generation. We will also expand the chemical vocabulary to include post-translational modifications, such as cyclization, to better support therapeutic applications.

7. Acknowledgments

This work was supported by NIH R01 award (R01GM143370). S.T.S. received partial support from an NIH R35 Award (GM147579). The agentic work flow computation was supported by Google Cloud Research Credits program with the award GCP19980904.

References

Abramson, J., Adler, J., Dunger, J., Evans, R., Green, T., Pritzel, A., et al. Accurate structure prediction of biomolecular interactions with alphafold 3. *Nature*, 630 (630):493–500, May 2024. doi: https://doi.org/10.1038/s41586-024-07487-w.

Bhadra, P., Yan, J., Li, J., Fong, S., and Siu, S. W. I. Ampep: Sequence-based prediction of antimicrobial peptides using distribution patterns of amino acid properties and random forest. *Scientific Reports*, 8(1), Jan 2018. doi: https://doi.org/10.1038/s41598-018-19752-w.

Boutet, E., Lieberherr, D., Tognolli, M., Schneider, M., and Bairoch, A. Uniprotkb/swiss-prot. *Methods in molecular biology (Clifton, N.J.)*, 406:89–112, Jan 2007. doi: https://doi.org/10.1007/978-1-59745-535-0_4. URL https://europepmc.org/article/MED/18287689.

Bucataru, C. and Ciobanasu, C. Antimicrobial peptides: Opportunities and challenges in overcoming resistance. *Microbiological Research*, 286:127822, Jun 2024. doi: https://doi.org/10.1016/j.micres.2024.127822.

Camacho, C., Coulouris, G., Avagyan, V., Ma, N., Papadopoulos, J., Bealer, K., and Madden, T. L. Blast+: architecture and applications. *BMC Bioinformatics*, 10 (1):421, 2009.

Fallahpour, A., Seyed-Ahmadi, A., Idehpour, P., Ibrahim, O., Gupta, et al. Bioreason-pro: Advancing protein function prediction with multimodal biological reasoning. *bioRxiv*, 2026. doi: 10.64898/2026.03.19.712954.

Fantner, G. E., Barbero, R. J., Gray, D. S., and Belcher, A. M. Kinetics of antimicrobial peptide activity measured on individual bacterial cells using high-speed atomic force microscopy. *Nature nanotechnology*, 5(4):280–285, Mar 2010. doi: https://doi.org/10.1038/nnano.2010.29.

Ferrell, J. B., Remington, J. M., Van, C. M., Sharafi, M., Aboushousha, R., Janssen-Heininger, Y., Schneebeli, S. T., Wargo, M. J., Wshah, S., and Li, J. A generative approach toward precision antimicrobial peptide design. *bioRxiv (Cold Spring Harbor Laboratory)*, Oct 2020. doi: https://doi.org/10.1101/2020.10.02.324087.

Gargate, N., Laws, M., and Rahman, K. M. Current economic and regulatory challenges in developing antibiotics for gram-negative bacteria. *npj Antimicrobials and Resistance*, 3(1), Jun 2025. doi: https://doi.org/10.1038/s44259-025-00123-1.

Google Cloud. Gemini 3.1 flash. https://docs.cloud.google.com/vertex-ai/generative-ai/docs/models/gemini/3-1-flash-lite, 2026a. Accessed: 2026-05-05.

Google Cloud. Gemini 3.1 pro. https://docs.cloud.google.com/vertex-ai/generative-ai/docs/models/gemini/3-1-pro, 2026b. Accessed: 2026-05-05.

Kossale, Y., Airaj, M., and Darouichi, A. Mode collapse in generative adversarial networks: An overview. In *2022 8th International Conference on Optimization and Applications (ICOA)*, pp. 1–6, 2022. doi: 10.1109/ICOA55659.2022.9934291.

Lin, Z., Akin, H., Rao, R., Hie, B., Zhu, Z., Lu, W., Smetanin, N., Verkuil, R., Kabeli, O., Shmueli, Y., dos Santos Costa, A., Fazel-Zarandi, M., Sercu, T., Candido, S., and Rives, A. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 379(6637):1123–1130, Mar 2023. doi: https://doi.org/10.1126/science.ade2574.

Lucana, M. C., Arruga, Y., Petrachi, E., Roig, A., Lucchi, R., and Oller-Salvia, B. Protease-resistant peptides for targeting and intracellular delivery of therapeutics. *Pharmaceutics*, 13(12), 2021. ISSN 1999-4923. doi: 10.3390/pharmaceutics13122065. URL https://www.mdpi.com/1999-4923/13/12/2065.

Ma, X., Wang, Q., Ren, K., Xu, T., Zhang, Z., Xu, M., Rao, Z., and Zhang, X. A review of antimicrobial peptides: Structure, mechanism of action, and molecular optimization strategies. *Fermentation*, 10(11):540–540, Oct 2024. doi: https://doi.org/10.3390/fermentation10110540.

Murray, C. J. L., Ikuta, K. S., Sharara, F., Swetschinski, L., Aguilar, G. R., et al. Global burden of bacterial antimicrobial resistance in 2019: a systematic analysis. *The Lancet*, 399(10325):629–655, Feb 2022. doi: https://doi.org/10.1016/S0140-6736(21)02724-0.

Panapitiya, G., Saldanha, E., Job, H., and Hess, O. Autolabs: Cognitive multi-agent systems with self-correction for autonomous chemical experimentation. *arXiv*, 2025. doi: https://doi.org/10.48550/arXiv.2509.25651.

Perez, E., Strub, F., de Vries, H., Dumoulin, V., and Courville, A. C. Film: Visual reasoning with a general conditioning layer. In *AAAI*, 2018.

Pirtskhalava, M., Amstrong, A. A., Grigolava, M., Chubinidze, M., Alimbarashvili, E., Vishnepolsky, B., Gabrielian, A., Rosenthal, A., Hurt, D. E., and Tartakovsky, M. Dbaasp v3: database of antimicrobial/cytotoxic activity and structure of peptides as a resource for development of new therapeutics. *Nucleic Acids Research*, 49(D1): D288–D297, Nov 2020. doi: https://doi.org/10.1093/nar/gkaa991. URL https://academic.oup.com/nar/article/49/D1/D288/5957160.

Ponnapati, M., Cox, S., Gordon, C. W., Hammerling, M. J., Narayanan, S., Laurent, J. M., Braza, J. D., Hinks, M. M., Skarlinski, M. D., Rodriques, S. G., and White, A. Proteincrow: A language model agent that can design proteins. In *ICML 2025 Generative AI and Biology (GenBio) Workshop*, 2025. URL https://openreview.net/forum?id=ljXgWDtqCu.

Schrödinger, LLC. *Maestro, Schrödinger, LLC Release 2026-2*. New York, NY, 2025. Accessed: 2026-03-01.

Silver, L. L. Challenges of antibacterial discovery. *Clinical microbiology reviews*, 24(1):71–109, 2011. doi: https://doi.org/10.1128/CMR.00030-10.

Szymczak, P., Możejko, M., Grzegorzek, T., Jurczak, R., Bauer, M., Neubauer, D., Sikora, K., Michalski, M., Sroka, J., Setny, P., Kamysz, W., and Szczurek, E. Discovering highly potent antimicrobial peptides with deep generative model hydramp. *Nature Communications*, 14(1), Mar 2023. doi: https://doi.org/10.1038/s41467-023-36994-z.

Tucs, A., Tran, D. P., Yumoto, A., Ito, Y., Uzawa, T., and Tsuda, K. Generating ampicillin-level antimicrobial peptides with activity-aware generative adversarial networks. *ACS Omega*, 5(36):22847–22851, 2020. doi: 10.1021/acsomega.0c02088.

Van Oort, C. M., Ferrell, J. B., Remington, J. M., Wshah, S., and Li, J. Ampgan v2: Machine learning-guided design of antimicrobial peptides. *Journal of Chemical Information and Modeling*, 61(5):2198–2207, Mar 2021. doi: https://doi.org/10.1021/acs.jcim.0c01441.

Veltri, D., Kamath, U., and Shehu, A. Deep learning improves antimicrobial peptide recognition. *Bioinformatics*, 34(16):2740–2747, Mar 2018. doi: https://doi.org/10.1093/bioinformatics/bty179.

Wang, G., Schmidt, C., Li, X., and Wang, Z. Apd6: the antimicrobial peptide database is expanded to promote research and development by deploying an unprecedented information pipeline. *Nucleic Acids Research*, 54(D1): D363–D374, 01 2026. ISSN 1362-4962. doi: 10.1093/nar/gkaf860. URL https://doi.org/10.1093/nar/gkaf860.

Wang, J., Feng, J., Kang, Y., Pan, P., Ge, J., Wang, Y., Wang, M., Wu, Z., Zhang, X., Yu, J., Zhang, X., Wang, T., Wen, L., Yan, G., Deng, Y., Shi, H., Hsieh, C.-Y., Jiang, Z., and Hou, T. Discovery of antimicrobial peptides with notable antibacterial potency by an llm-based foundation model. *Science Advances*, 11(10), Mar 2025a. doi: https://doi.org/10.1126/sciadv.ads8932.

Wang, L., Fu, X., Yang, J., Zhang, X., Ye, X., Sakurai, T., Zeng, X., and Liu, Y. Moformer: navigating the antimicrobial peptide design space with pareto-based multiobjective transformer. *Briefings in Bioinformatics*, 26(6), Nov 2025b. doi: https://doi.org/10.1093/bib/bbaf376.

Wang, L., Liu, Y., Fu, X., Ye, X., Shi, J., Yen, G. G., Zou, Q., Zeng, X., and Cao, D. Hmamp: Designing highly potent antimicrobial peptides using a hypervolume-driven multiobjective deep generative model. *Journal of Medicinal Chemistry*, 68(8):8346–8360, 2025c. doi: 10.1021/acs.jmedchem.4c03073. PMID: 40232176.

Wang, Y., Lu, J., Jaitly, N., Susskind, J., and Bautista, M. A. Simplefold: Folding proteins is simpler than you think. In *ICLR*, 2025d. URL https://arxiv.org/abs/2509.18480v1.

WHO. Antimicrobial resistance, Nov 2023. URL https://www.who.int/news-room/fact-sheets/detail/antimicrobial-resistance.

Yakimov, A., Afanaseva, A., Khodorkovskiy, M., and Petukhov, M. Design of stable -helical peptides and thermostable proteins in biotechnology and biomedicine. *Acta Naturae*, 8(4):70–81, Dec 2016. doi: https://doi.org/10.32607/20758251-2016-8-4-70-81.

Zhang, Z. Introduction to machine learning: k-nearest neighbors. *Annals of Translational Medicine*, 4(11):218–218, Jun 2016. doi: https://doi.org/10.21037/atm.2016.03.37.

A. In Vitro Experimental Protocol

Selected peptide candidates were commercially synthesized by a MedChemExpress(MCE) using standard solid-phase peptide synthesis and purified by high-performance liquid chromatography (HPLC). Peptide identity and purity were confirmed by mass spectrometry and analytical HPLC according to the quality-control reports.

Table 7. Bacterial isolates used in the current study.

Bacteria	Strain ID	Strain Designation	Source
S. aureus	USA300-0114	NRD384, MRSA	Wound, Mississippi, USA
B. subtilis	—	—	—
S. epidermidis	ATCC 35984	NRS 101	Catheter sepsis, Tennessee, USA
E. faecium	HM-952	503	Human, USA
E. coli	ATCC 25922	Seattle 1946	Washington, USA

B. AMPGAN v3: Implementation Details and Additional Analyses

All training was conducted on an NVIDIA RTX PRO 6000 Blackwell GPU, and all inference was performed on an NVIDIA GeForce RTX 4080.

B.1. Note on D- vs L-Amino Acids in Evaluation

A central contribution of AMPGAN v3 is the ability to generate sequences containing non-canonical residues (D-amino acids) and N/C-terminus modifications. We emphasize a distinction between two settings in this paper:

- **In vitro evaluation (Section 5.1).** Candidates synthesized for wet-lab testing were drawn from the *full* generative vocabulary, including D-amino acid substitutions and C-terminal amidation, since these modifications are indispensable for therapeutic viability (Lucana et al., 2021). This setting reflects the intended use case of AMPGAN v3.
- **In silico evaluation (Section 5.2 and Tables 4–5).** For all quantitative comparisons against AMPGAN v2, HydrAMP, and AMP-Designer, we generate sequences using the *L-amino acid subset* of our vocabulary only. None of the baseline models support D-amino acids or terminal modifications, and the external classifiers (AMP Scanner v2, amPEP) and embedding models (ESM2) used for evaluation are trained exclusively on canonical L-peptides. Including non-canonical residues in the comparison would have introduced an out-of-distribution artifact that confounds rather than informs the comparison. Restricting to L-only generations ensures the gains we report reflect modeling improvements rather than vocabulary differences.

B.2. Hyperparameters and Training Configuration

Table 8. AMPGAN v3 training hyperparameters.

Generator		**Discriminators**	
Latent dim d_z	256	d_{model}	64
Condition embed d_e	128	FFN hidden d_{hid}	256
Transformer L / H	4 / 4	Transformer L / H	4 / 4
Hidden dim	128	MLP head dim	64
Gumbel temp τ	1.0	Dropout	0.2
Loss weights		**Optimization**	
λ_ℓ (length)	0.5	Optimizer	AdamW, β=$(0.5, 0.999)$, wd 10^{-4}
λ_s (similarity)	0.5	LR (G / D_{adv} / D_{mic})	10^{-4} / 5×10^{-6} / 10^{-4}
λ_a (adversarial)	2	Batch size	256
λ_m (MIC)	1	Epochs	200
α (padding)	0.1	D : G updates	1 : 1

B.3. Token Vocabulary

Each sequence is encoded as `[N-term][SOS][Amino Acid Tokens][EOS][C-term]`, with maximum length $L = 64$ residues. The full vocabulary (Table 9) contains $V = 48$ tokens spanning canonical L-amino acids, D-amino acid stereoisomers, sequence-boundary markers, padding, and N/C-terminus chemical modifications. Lowercase letters denote D-amino acids, preserving stereochemistry directly in the token stream. This unified representation allows AMPGAN v3 to jointly generate sequence content and chemical modifications within a single output stream, in contrast to prior generative AMP models that operate over the 20 canonical L-amino acids only.

Table 9. AMPGAN v3 token vocabulary. Lowercase tokens denote D-amino acid stereoisomers of the corresponding uppercase L-amino acid. `<C8>`–`<C16>` denote fatty-acid acylation at the N-terminus with the indicated chain length.

Category	Tokens
L-amino acids (20)	`A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y`
D-amino acids (16)	`a, c, f, h, i, k, l, n, p, q, r, s, t, v, w, y`
Sequence boundaries	`<SOS>`, `<EOS>`
Sequence padding	`<blank>`
N-terminus modifications	`<ACT>` (acetylation), `<C8>`, `<C10>`, `<C12>`, `<C14>`, `<C16>`, `<nblank>`
C-terminus modifications	`<AMD>` (amidation), `<cblank>`

C. Agentic Pipeline: Architecture, Prompts, and Trajectories

C.1. System Prompts

Planning Agent.

```
You are a **Planning Agent** orchestrating a multi-agent system for
**Antimicrobial Peptide (AMP) Discovery**.
Your role is to coordinate the workflow by selecting the next agent and issuing
structured instructions in strict JSON format.
Available Agents
{Agent Description}
{Agent Context} ---
Planning Logic - If no peptide sequences exist, you MUST generate sequences first.
- If peptide sequences exist but are unfiltered, you MUST filter.  - If filtered
results satisfy all constraints, you MUST select **END**.  - You MAY iterate
between Agents to refine results.
---
Output Rules (STRICT) - Output ONLY JSON - No explanations, no extra text - Must
follow schema exactly - Do NOT assume hidden state | rely only on explicitly
provided input context
---
JSON Format
{
  "Planning":  {
    "Agent":  "Generating | Filtering | Verifying | END",
    "Purpose":  "2--3 sentence description of the current objective for the
selected agent.",
    "Context":  {
      "instructions":  "Provide clear, structured instructions for the selected
agent to complete the assigned task.  This should describe what to do, not raw
input data.",
      "details":  "Contains all execution-critical information such as
user-provided inputs, parameters, constraints, tool-call arguments, or any
supplemental data required for the agent to perform its function accurately."
    }
  }
}
```

Execution Agent.

```
You are a **{Agent} Agent** designed to execute tool-based biological workflows for
**Antimicrobial Peptide (AMP)** tasks.  Your role is to follow the instructions
provided by the **Planning Agent** (your supervisor) and perform concrete actions
using appropriate biological tools.

Context - **Planning Agent Instruction:**

{Instruction}

- **Your Capabilities:**

{Agent Description}

Rules 1.  Output ONLY valid JSON matching the schema below | no prose, no
explanation.  2.  Each step is exactly one tool invocation.  Steps are numbered
sequentially from 1.  3.  Use only tools listed in your Agent Description.  All
required parameters must be provided.  4.  You may call the same tool multiple
times if needed.  5.  You may add calls to reporting/support tools from your
description if they improve output quality, but never invent tools.

Output Schema

{
  "Agent":  "{Agent}",
  "Objective":  "<concise task summary>",
  "Steps":  [
    {
      "id":  1,
      "Tool":  "<tool_name>",
      "Input":  { "<param>":  "<value>" }
    }
  ]
}
```

C.2. Example Outputs

Planning Agent.

```
{
  "Planning":  {
    "Agent":  "Generating",
    "Purpose":  "Generate the initial batch of Antimicrobial Peptide
sequences targeting E. coli using AMPGAN-v3.",
    "Context":  {
      "instructions":  "Use the AMPGAN_v3 tool to generate an initial batch
of sequences targeting E. coli.  Generate exactly 5 sequences for the
first pass, with lengths between 10 and 20 amino acids.",
      "details":  {
        "min_length":  10,
        "max_length":  20,
        "num_generations":  5,
        "species_of_interest":  "ecoli",
        "folder_path":  "PATH"
      }
    }
  }
}
```

Execution Agent.

```
{
  "Agent": "Verifying",
  "Objective": "Cross-reference filtered AMP candidates against
SwissProt and DBAASP databases to assess novelty and retrieve biological
annotations.",
  "Steps": [
    {
      "id": 1,
      "Tool": "Verify_SwissProt",
      "Input": {
        "folder_path": "PATH"
      }
    },
    {
      "id": 2,
      "Tool": "Verify_DBAASP",
      "Input": {
        "folder_path": "PATH"
      }
    }
  ]
}
```

C.3. Executor Report Example

```
**Execution Overview** The objective of this workflow was to refine a set
of generated antimicrobial peptides (AMPs) by applying a sequence of four
specific filters: length, cationicity, hydrophobicity, and D-amino acid
composition. These filters were applied to a dataset located at "PATH" to
identify high-quality candidates.

**Step-wise Results** The length filter was applied to select sequences
between 10 and 20 amino acids, followed by a cationicity filter (2.0 -
8.0) and a hydrophobicity filter (-0.5 - 0.5). Finally, a D-amino acid
filter was executed. For each of these four steps, the original CSV
file was updated with corresponding indicator columns ('length_filter',
'cationicity_filter', 'hydrophobicity_filter', and 'damino_filter'), where
a value of 1 marks a successful pass and 0 denotes a failure to meet the
criteria.

**Final Outcome** The final status assessment confirms that, from the
initial pool of 5 generated sequences, 1 sequence successfully passed
all applied filters. This sequence met the requirements for length,
cationicity, hydrophobicity, and D-amino acid composition, representing
the single candidate remaining after the full filtering process.

**Completed Tasks** The successfully completed tasks include the Length
Filter, Cationicity Filter, Hydrophobicity Filter, D-amino acid Filter,
and the final Filter Status report.
```

C.4. Tool Specifications

Each executor agent operates over a fixed set of tools registered in its system prompt. Tools accept structured JSON inputs from the agent and output string-template reports describing their outcome (e.g., counts of surviving candidates, BLAST hits, generation summaries). The executor agent aggregates these per-tool reports into a single run report, which is routed back to

the Planning Agent as context for its next decision. This separation lets tools remain stateless and replaceable while the executor handles report composition. Below we list each agent's tools, their inputs, and operational notes that the Planning Agent uses when deciding whether to re-invoke.

Generating Agent. Translates user-defined natural-language constraints into structured tool inputs.

- `AMPGAN_v3` — GAN-based AMP generator (Section 3). Outputs are written to `generated_sequences.csv` for downstream tools. *Inputs*: `min_length`, `max_length` (int), `num_generations` (int), `species_of_interest` (str; one of `ecoli`, `paeruginosa`, `kpneumoniae`, `saureus`, `bsubtilis`, `sepidermidis`), `folder_path`.
- `Predict_Structure` — 3D structure prediction via SimpleFold (Wang et al., 2025d) 360M. Used only on filtered subsets to bound cost; the Planning Agent is instructed to defer this tool until after physicochemical filtering. *Inputs*: `folder_path`.

The Generating Agent follows an adaptive batch-sizing policy: the first invocation generates the user-requested count N; after observing the post-filter pass rate p, subsequent rounds generate $\lceil S/p \rceil$ candidates to cover the shortfall S, capped at $3N$ per call.

Filtering Agent. Applies sequential physicochemical and structural screens. Each filter writes a boolean column to the candidate CSV and the Filter_Status tool reports the surviving count. The Planning Agent prioritizes cheap property filters before invoking structure-dependent filters.

- `Cation_Filter` — Filters by net charge at pH 7. *Inputs*: `min_cationicity` (float), `max_cationicity` (float), `folder_path`.
- `Hydrophobicity_Filter` — Filters by GRAVY score, balancing membrane insertion against aqueous solubility. *Inputs*: `min_hydrophobicity` (float), `max_hydrophobicity` (float), `folder_path`.
- `Length_Filter` — Filters by amino acid count. *Inputs*: `min_length` (int), `max_length` (int), `folder_path`.
- `Structure_Filter` — Classifies predicted PDB structures into one of six classes: mixed, α-helical, β-hairpin, extended β-strand, structured turns, or unstructured. Requires PDB output from `Predict_Structure`; does not perform folding itself. The `structure` input is a semicolon-separated string of class IDs(e.g., `"2;3"`). *Inputs*: `folder_path`, `structure` (str).
- `Damino_Filter` — Retains sequences containing at least one D-amino acid (lowercase letter in the token stream). Invoked when the user objective specifies non-canonical chemistry. *Inputs*: `folder_path`.
- `Filter_Status` — Mandatory bookkeeping tool. Reports the integer count of surviving candidates after each filtering cycle. The Planning Agent uses this output to decide between proceeding, re-filtering, and re-generating.*Inputs*: `folder_path`.

Verifying Agent. Cross-references filtered candidates against external databases to assess novelty and biological context. Both BLAST searches return the top hit per candidate within an E-value threshold of 10. While elevated E-value cutoffs are standard for short-peptide BLAST searches, the threshold here is set permissively for demonstration and is exposed as a tunable parameter.

- `Verify_SwissProt` — Local BLAST (Camacho et al., 2009) alignment against SwissProt (Boutet et al., 2007), returning top homologs with taxonomic classifications and cross-referenced annotations. *Inputs*: `folder_path`.
- `Verify_DBAASP` — BLAST alignment against the DBAASP (Pirtskhalava et al., 2020) subset used to train AMPGAN v3, returning the closest training-set match together with reported antimicrobial activity. Used to assess training-set proximity and recover known-activity annotations for near-duplicates. *Inputs*: `folder_path`.

C.5. Representative Trajectory

We include a representative trajectory from a multi-agent run with objective "Generate 5 AMPs with D amino acid targeting *E. coli* with α-helical secondary structure."

1	**Planning Agent → Generating Agent** Execute AMPGAN_v3 to generate a batch of AMP sequences based on the user's initial request. Ensure length constraints and target species are applied.	# of Generated: 5 # of Total Generated: 5
2	**Planning Agent → Filtering Agent** Execute the length, cationicity, hydrophobicity, and D-amino acid filters on the generated sequences. Once all preliminary filters are applied, use the Filter_Status tool to report the current count of surviving candidates.	# of Filtered: 1
3	**Planning Agent → Generating Agent** Use AMPGAN_v3 to generate 20 more peptide sequences targeting E. coli. This calculation is based on needing 4 more candidates and observing a 20% pass rate in the initial filtering round.	# of Generated: 20 # of Total Generated: 25
4	**Planning Agent → Filtering Agent** Run Damino_Filter to ensure D-amino acids are present, followed by Length_Filter, Cation_Filter, and Hydrophobicity_Filter. Conclude with Filter_Status to report the number of surviving candidates.	# of Filtered: 3
5	**Planning Agent → Generating Agent** Run AMPGAN_v3 to generate a second batch of sequences. Based on the previous pass rate (3 out of 25 = 12%), generate 17 additional sequences targeting E. coli to compensate for the shortfall of 2 candidates.	# of Generated: 17 # of Total Generated: 42
6	**Planning Agent → Filtering Agent** Execute the fundamental property checks on the updated sequence pool. Apply the length filter, cationicity filter, hydrophobicity filter, and D-amino acid filter. Finally, invoke the Filter_Status tool to report the total count of candidates that have survived these preliminary screens.	# of Filtered: 3
7	**Planning Agent → Generating Agent** Use AMPGAN_v3 to generate a supplemental batch of sequences targeting Ecoli. Calculate the number of sequences to generate based on the shortfall and observed pass rate, but strictly cap this generation call at 3 times the requested count (15 sequences) to bound costs.	# of Generated: 15 # of Total Generated: 57
8	**Planning Agent → Filtering Agent** Execute the fundamental property checks on the updated dataset. Apply the length, cationicity, hydrophobicity, and D-amino acid filters to isolate sequences meeting the user's constraints. Conclude by calling the Filter_Status tool to report the surviving candidate count.	# of Filtered: 5
9	**Planning Agent → Generating Agent** Execute the Predict_Structure tool on the refined subset of 5 sequences to generate PDB files. This step is strictly required before the structural classification filter can isolate alpha-helical conformations.	
10	**Planning Agent → Filtering Agent** Execute the Structure_Filter tool to classify and select peptides that form an alpha-helical conformation based on the recently predicted 3D structures. Afterward, run the Filter_Status tool to report the current count of surviving candidates.	# of Filtered: 5
11	**Planning Agent → Verifying Agent** Execute the Verify_SwissProt tool to cross-reference the 5 successfully filtered candidates against the protein database. Retrieve the closest matching sequences, their taxonomic classifications, and relevant annotations.	