TINAS: Transformer Interaction among N Agents with Shared Parameters

Kasra Fouladi
Iran University of Science and Technology (IUST)
k4sr405@gmail.com
https://github.com/kasrafouladi/StrikeForce-No-BPTT-via-FAP

Repository: https://github.com/kasrafouladi/StrikeForce-No-BPTT-via-FAP

Abstract

We study inter-agent communication in multi-agent reinforcement learning (MARL) with parameter sharing. Standard PPO treats each agent's loss as local: no mechanism lets agents learn to communicate in a gradient-aware way. We propose TINAS, a framework in which each agent emits a message to every other agent at each timestep, and messages are trained by *future-gradient injection*. The sender's message loss at time t is built from the receiver's loss at time $t+1$, differentiated with respect to the sender's positional encoding; because the token construction is linear in both the raw message and the positional encoding, this gradient equals the true gradient of the receiver's future loss with respect to the message, and no re-attachment step is required. The resulting algorithm is, to our knowledge, the first multi-agent communication-learning method that achieves a T-step credit-assignment horizon at autograd graph depth $\Theta(1)$ per step, with a bounded, practitioner-controlled accuracy cost. The detached objective is a *semi-gradient*: it is not the gradient of any fixed function, so we analyze stationarity against the attached reference $\mathcal{L}^*$ that a fully-differentiable method such as DIAL would optimize. Under standard smoothness and boundedness assumptions, TINAS converges to a $2\mathcal{B}^2$-neighborhood of a stationary point of $\mathcal{L}^*$ at rate $C_1/\sqrt{K}$, where $\mathcal{B}$ is a straight-through-estimator bias bound controlled by c_m and the message magnitude. The two-pass backward algorithm attains this at $O(T)$ cost with no buffer replay, and we instantiate TINAS on a frozen behavioral-cloning backbone for the StrikeForce environment.

Domain: Multi-Agent Reinforcement Learning, Communication, Optimization
Keywords: multi-agent RL, parameter sharing, communication learning, straight-through estimator, future-gradient injection

1 Introduction

1.1 Problem

Multi-agent reinforcement learning with parameter sharing is a powerful paradigm: a single policy network shared across all agents avoids the combinatorial blowup of independent learning and lets agents generalize across roles (Foerster et al., 2016; Sukhbaatar et al., 2016). Yet standard PPO training (Schulman et al., 2017) treats each agent's loss as local—no mechanism lets agents *learn to communicate* in a gradient-aware way.

1.2 Why Naive Communication Fails

A natural remedy is to add a communication head that emits messages between agents and backpropagate through the receiver's loss into the sender. This is essentially what DIAL (Foerster et al., 2016) does. However, differentiating through a T-step rollout with attached messages requires storing the full computational

graph for every timestep, which is $O(T)$ memory and incompatible with standard PPO minibatching. Loosening the graph (detach the message, then re-attach it before the receiver's backward) breaks the gradient chain and yields a biased estimator that is silent about its own bias.

1.3 Our Approach: Future-Gradient Injection

TINAS closes this gap with a differentiable inter-agent communication channel built on a shared transformer backbone. Each agent i emits a message $m^t_{i\to j}$ to every other agent j at each timestep. The messages are trained by *future-gradient injection*: the sender's message loss at time t is built from the receiver's loss at time $t+1$, differentiated with respect to the sender's positional encoding $\text{pos}[i]$. Because the token construction is linear in both the raw message and $\text{pos}[i]$, this gradient equals the true gradient of the receiver's future loss with respect to the message. Gradient injection is therefore possible *without any re-attachment step.*

1.4 Summary of Contributions

- **A gradient-equivalence lemma (Lemma 1).** For a token construction linear in the message and in the sender's positional encoding, $\partial L_j^{t+1}/\partial m^t_{i\to j} = \partial L_j^{t+1}/\partial \text{pos}[i]$ exactly. The injected gradient is extracted with a single `autograd.grad` call on the positional encoding; no re-attachment is needed.
- **A two-pass backward algorithm (Section 4).** Pass 1 records detached snapshots under `NoGradGuard`. Pass 2 reconstructs each step's loss, extracts the injected gradient, and accumulates parameter gradients. Cost is $O(T)$, the same asymptotic scaling as standard PPO, with no buffer replay.
- **A convergence analysis against a stationary-point reference (Section 3).** Because the detached objective is a semi-gradient, we analyze it against the attached reference $\mathcal{L}^*$ that a fully-differentiable method would optimize. The injected-gradient recurrence is contractive when $c_m L_{\text{net}} < 1$ (Proposition 4); the STE bias is bounded by $\mathcal{B} = c_m \sup_t \|m^t\| B_2$ (Lemma 5); SGD on the semi-gradient converges to a $2\mathcal{B}^2$-neighborhood of a stationary point of $\mathcal{L}^*$ at rate $C_1/\sqrt{K}$ (Theorem 7); and $\mathcal{B} \to 0$ recovers the exact BPTT gradient (Corollary 6).
- **Credit assignment at bounded graph depth (Section 3.8).** TINAS achieves a credit-assignment horizon of $T - t$ steps at autograd graph depth $\Theta(1)$ per step. This is a structural property of the algorithm, not a claim about any specific alternative mechanism. The price is the bounded second-order bias of Lemma 5. Among methods that propagate the gradient through a full T-step computational graph, TINAS reduces activation memory by a factor of $\Omega(L)$, where L is the transformer depth (Table 2).
- **A concrete instantiation (Section 9)** on a shared-parameter behavioral-cloning backbone for the StrikeForce 2D battle-arena testbed.

2 Model

2.1 Notation

We consider n agents. The communication extension has parameter vector $\theta \in \mathbb{R}^d$; the pretrained backbone is held fixed. A rollout has T timesteps $t = 1, \ldots, T$. At each step, agent i sends a message $m^t_{i\to j}$ to each other agent $j \neq i$. Two positional embeddings are used: $\text{pos}[i] \in \mathbb{R}^{d_{\text{msg}}}$ for each sender i, and a spatial distance vector $d^t[j \to i] \in \mathbb{R}^{d_{\text{dist}}}$ encoding the geometry between receiver j and sender i.

Boundary condition. $L_i^{T+1} = 0$ (constant, for all i).

Per-agent per-step loss.

$$L_i^t \;=\; L_{\text{PPO}}^{i,t} + c_V L_V^{i,t} - c_H H^{i,t} + \frac{c_m}{n-1} \sum_{j \neq i} L^t_{m,i\to j}.$$

Message loss.

$$L^t_{m,i\to j} \;=\; \big(g^{t+1}_{i\to j}\big)^\top m^t_{i\to j}.$$

Injected gradient.

$$g^{t+1}_{i\to j} \;=\; \Big(\frac{\partial L^{t+1}_j}{\partial \mathrm{pos}[i]}\Big)_{\mathrm{detached}}.$$

Extracted with `create_graph=False`; carries no computational graph.

Total objective.

$$\mathcal{L}(\theta) \;=\; \frac{1}{Tn}\sum_{t=1}^{T}\sum_{i=1}^{n} L^t_i.$$

Attached reference objective. The *attached* objective $\mathcal{L}^*$ is the same expression as $\mathcal{L}$ but with g^{t+1} not detached. It is the objective that a fully-differentiable method such as DIAL would optimize in the same architectural setup. Since $\mathcal{L}$ treats g^{t+1} as a constant in θ whereas g^{t+1} depends on θ through m^t, the autograd derivative $\tilde{\nabla}\mathcal{L}$ of $\mathcal{L}$ is a *semi-gradient* in the sense of **?**: it is not the gradient of any fixed function. We therefore use $\mathcal{L}^*$ as the fixed reference against which convergence is stated, and we bound the gap $\tilde{\nabla}\mathcal{L} - \nabla\mathcal{L}^*$ in Lemma 5. Table 1 summarizes the three objects.

Symbol	Role in the analysis	Differentiable in θ?
$\mathcal{L}$	detached loss value; what the algorithm computes	value yes, autograd semi-gradient
$\mathcal{L}^*$	attached reference; fixed function of θ	yes (full gradient)
$\tilde{\nabla}\mathcal{L}$	the algorithm's semi-gradient	—

Table 1: Three related objects used in the analysis.

2.2 Token Construction (Exact)

For each ordered pair (i, j) with $i \neq j$, the message token from sender i to receiver j at step t is

$$\boxed{\mathrm{tok}^t_{i\to j} \;=\; \mathrm{proj}\big(\mathrm{cat}\big\{m^t_{i\to j} + \mathrm{pos}[i],\; d^t[j \to i]\big\}\big)\,.}$$

Here $\mathrm{proj} : \mathbb{R}^{d_{\mathrm{msg}}+d_{\mathrm{dist}}} \to \mathbb{R}^{d_{\mathrm{model}}}$ is a linear map (the `msg_proj` module), and `cat` denotes concatenation. The first block is $m^t_{i\to j} + \mathrm{pos}[i]$; the second is the spatial distance vector.

At receiver j, the messages from all senders $i \neq j$ enter the transformer's input together with j's own r^t_j and a learnable value token v. After the transformer, the message slots are projected by a linear head to produce outbound $m^t_{j\to k}$.

Remark 1 (Why the linearity matters)**.** *The composition is linear in both $m^t_{i\to j}$ and $pos[i]$. A non-linear projection, or a token construction that mixed $pos[i]$ into multiple slots, would break the equivalence of Lemma 1.*

2.3 Gradient Equivalence

Lemma 1 (Gradient equivalence)**.** *For any $i \neq j$ and any t,*

$$\frac{\partial L^{t+1}_j}{\partial m^t_{i\to j}} \;=\; \frac{\partial L^{t+1}_j}{\partial pos[i]}$$

as tensors in $\mathbb{R}^{d_{msg}}$.

Proof. Let J_{proj} be the Jacobian of proj. Write $a = m^t_{i\to j} + \text{pos}[i]$. Then

$$\frac{\partial \text{tok}^t_{i\to j}}{\partial m^t_{i\to j}} \;=\; J_{\text{proj}}[:, 0{:}d_{\text{msg}}] \;=\; \frac{\partial \text{tok}^t_{i\to j}}{\partial \text{pos}[i]}.$$

By the chain rule through L^{t+1}_j, the two gradients are equal. □

Remark 2 (Dependence of L^{t+1}_j on (i, j))**.** *L^{t+1}_j depends on $m^t_{i\to j}$ only through the single token $(i \to j)$ at receiver j. Each sender contributes exactly one token per receiver; the constant gather index* `other_idx[i]`$= [j \neq i]$ *enforces this.*

Remark 3 (No re-attachment required)**.** *The algorithm extracts $g^{t+1}_{i\to j}$ via* `autograd.grad`(L^{t+1}_j, pos) *rather than* `autograd.grad`(L^{t+1}_j, m^t)*. This is exact: $pos[i]$ is attached (a parameter), while $m^t_{i\to j}$ is detached on the send side.*

2.4 Per-Step Loss (Restated)

$$L^t_i \;=\; L^{i,t}_{\text{PPO}} + c_V L^{i,t}_V - c_H H^{i,t} + \frac{c_m}{n-1} \sum_{j\neq i} \left(g^{t+1}_{i\to j}\right)^\top m^t_{i\to j}.$$

2.5 Total Objective

$$\mathcal{L}(\theta) \;=\; \frac{1}{Tn} \sum_{t=1}^{T} \sum_{i=1}^{n} L^t_i.$$

The training update is $\theta_{K+1} = \text{AdamW}(\theta_K, \hat{G}_K, \eta, \lambda)$ with $\hat{G}_K$ a minibatch estimate of $\tilde{\nabla}\mathcal{L}$ (the algorithm's semi-gradient).

3 Theoretical Analysis

Reading guide. Sections 1–5 state the assumptions. Lemma 2 establishes well-definedness. Lemma 3 derives the semi-gradient. Proposition 4 proves contractivity. Lemma 5 bounds the STE bias. Corollary 6 shows the exact-BPTT limit. Theorem 7–8 give the convergence rate. Section 3.8 characterizes credit assignment at bounded graph depth.

3.1 Assumptions

Assumption 1 (Smoothness of $\mathcal{L}^*$)**.** *$\mathcal{L}^* : \mathbb{R}^d \to \mathbb{R}$ is L_{lip}-smooth: $\|\nabla\mathcal{L}^*(\theta) - \nabla\mathcal{L}^*(\theta')\| \leq L_{lip}\|\theta - \theta'\|$. This is standard for networks composed of Lipschitz layers with bounded weights, and is enforced in practice by weight decay.*

Assumption 2 (Bounded update direction)**.** *The clipped gradient $\hat{G}^{clip} = \hat{G}/\max(1, \|\hat{G}\|)$ satisfies $\|\hat{G}^{clip}\| \leq 1$. Per-step update satisfies $\|\theta_{k+1} - \theta_k\| \leq \eta$. Used in Section 5; Theorem 7 does not require it directly.*

Assumption 3 (Bounded stochasticity)**.** *$\hat{G}_K = \tilde{\nabla}\mathcal{L}(\theta_K) + \xi_K$ with $\mathbb{E}[\xi_K \mid \theta_K] = 0$ and $\mathbb{E}[\|\xi_K\|^2 \mid \theta_K] \leq \sigma^2$, where $\tilde{\nabla}\mathcal{L}$ is the algorithm's semi-gradient (Lemma 3).*

Assumption 4 (Network Lipschitz)**.** *$L_{net} := \sup_\theta \max_{i,j,k,t} \|\partial m^t_{j\to k}/\partial pos[i]\|_{op} < \infty$.*

Assumption 5 (Bounded second derivative)**.** *$\sup_t \|\partial g^t/\partial\theta\| \leq B_2$ along the trajectory.*

A1–A3 are standard in nonconvex optimization (Ghadimi & Lan, 2013; Bottou et al., 2018). A4 controls contractivity; A5 controls the STE bias.

3.2 Well-Definedness

Lemma 2 (Recursion terminates). *$L^t_{m,i\to j}$ is well-defined for all $t \in [1,T]$ and $i \neq j$, and is zero at $t = T$.*

Proof. $L^{T+1}_j = 0 \Rightarrow g^{T+1}_{i\to j} = 0 \Rightarrow L^T_{m,i\to j} = 0$. By induction on $T-t$, every $L^t_{m,i\to j}$ has depth at most $T-t$ and at most $(n-1)^{T-t}$ terms. □

3.3 Gradient Computation

Lemma 3 (Semi-gradient formula). *The algorithm's semi-gradient is*

$$\tilde{\nabla}_\theta \mathcal{L}(\theta) \;=\; \frac{1}{Tn}\sum_{t,i}\Big[\nabla_\theta\big(L^{i,t}_{PPO} + c_V L^{i,t}_V - c_H H^{i,t}\big) + \frac{c_m}{n-1}\sum_{j\neq i}\big(g^{t+1}_{i\to j}\big)^\top \nabla_\theta m^t_{i\to j}\Big].$$

Proof. $g^{t+1}_{i\to j}$ is detached (constant in θ); the message-loss derivative is $(g^{t+1}_{i\to j})^\top \nabla_\theta m^t_{i\to j}$. □

3.4 Contractivity of the Recurrence

Define $T^t : \mathbb{R}^{n(n-1)\times d_{\text{msg}}} \to \mathbb{R}^{n(n-1)\times d_{\text{msg}}}$ by

$$(T^t G)_{(i,j)} = \frac{c_m}{n-1}\sum_{k\neq j} G_{(j,k)} \cdot \frac{\partial m^t_{j\to k}}{\partial \text{pos}[i]}.$$

The recurrence for the injected gradients is $g^t_{(i,j)} = b^t_{(i,j)} + (T^t g^{t+1})_{(i,j)}$, where $b^t_{(i,j)} = \partial(L^{j,t}_{\text{PPO}} + c_V L^{j,t}_V - c_H H^{j,t})/\partial \text{pos}[i]$ is the base term.

Proposition 4 (Contractivity). *Under A4, $\|T^t\|_{op} \leq c_m L_{net}$. When $c_m L_{net} < 1$, T^t is a strict contraction, and*

$$\|g^t\| \;\leq\; \sum_{s=0}^{T-t}(c_m L_{net})^s \|b^{t+s}\|.$$

Proof. For fixed (i,j), T^t sums $n-1$ terms with coefficient $c_m/(n-1)$, each bounded by $L_{\text{net}}\|G_{(j,k)}\|$. Row-wise dependence gives $\|T^t G\|_\infty \leq c_m L_{\text{net}}\|G\|_\infty$, dimension-free in n. □

Remark 4 (Role of $c_m/(n-1)$). *The $c_m/(n-1)$ factor makes the coefficient independent of n. Without it, the coefficient would be $c_m(n-1)L_{net}$.*

3.5 STE Bias

The injected gradient $g^{t+1}_{i\to j}$ is detached in $\mathcal{L}$ but attached in $\mathcal{L}^*$. The gap between the two gradient fields is bounded.

Lemma 5 (STE bias bound). *Under A5,*

$$\|\tilde{\nabla}_\theta \mathcal{L}(\theta) - \nabla_\theta \mathcal{L}^*(\theta)\| \;\leq\; \mathcal{B} \;:=\; c_m \sup_t \|m^t\| B_2.$$

Bias vanishes at $m \equiv 0$.

Proof. The difference is the missing second-order term $\frac{1}{Tn}\sum_{t,i,j}\frac{c_m}{n-1}(m^t_{i\to j})^\top \partial g^{t+1}_{i\to j}/\partial\theta$. Bounding each of the $Tn(n-1)$ terms by $c_m\|m\|B_2$ after normalization gives the result. □

Remark 5 (This is the price of graph-depth reduction). *The bias $\mathcal{B}$ is exactly the missing second-order term in the chain $m^t \to g^{t+1} \to \theta$. It is bounded, vanishes at $m = 0$, and scales linearly with c_m, so the trade-off between memory savings and gradient exactness is under the practitioner's control. Lower c_m shrinks $\mathcal{B}$ but slows the message learning signal; higher c_m accelerates message learning but inflates the floor.*

Corollary 6 (The exact-BPTT limit)**.** *As $\mathcal{B} \to 0$ (equivalently, $c_m \to 0$ or $\sup_t \|m^t\| \to 0$), $\tilde{\nabla}\mathcal{L} \to \nabla\mathcal{L}^*$ uniformly on bounded parameter sets, and the TINAS update approaches the exact backpropagation-through-time update of the same architectural objective.*

Proof. Immediate from Lemma 5: $\|\tilde{\nabla}\mathcal{L} - \nabla\mathcal{L}^*\| \le \mathcal{B} \to 0$. □

3.6 Convergence Theorem

Theorem 7 (Convergence to an approximate stationary point)**.** *Suppose A1, A3, A5 hold, and let $\mathcal{B}$ be the STE bias bound of Lemma 5. Run SGD with step size $\eta \in (0, 1/(4L_{lip})]$ on the semi-gradient, estimated by $\hat{G}_k$.*[1]*. Then for any $K \ge 1$:*

$$\min_{0\le k\le K-1} \mathbb{E}\big[\|\nabla\mathcal{L}^*(\theta_k)\|^2\big] \;\le\; \frac{4\Delta_0}{\eta K} + 2\mathcal{B}^2 + 4\eta L_{lip}(\mathcal{B}^2 + \sigma^2),$$

with $\Delta_0 := \mathcal{L}^*(\theta_0) - \inf \mathcal{L}^*$. Choosing $\eta = 1/\sqrt{K}$ and requiring $K \ge 16L_{lip}^2$:

$$\min_{0\le k\le K-1} \mathbb{E}\big[\|\nabla\mathcal{L}^*(\theta_k)\|^2\big] \;\le\; \frac{C_1}{\sqrt{K}} + 2\mathcal{B}^2, \quad C_1 := 4\Delta_0 + 4L_{lip}(\mathcal{B}^2 + \sigma^2).$$

Proof. By Lemma 5 and A3, $\hat{G}_k = \nabla\mathcal{L}^*(\theta_k) + b_k + \xi_k$ with $\|b_k\| \le \mathcal{B}$ and $\mathbb{E}[\xi_k \mid \theta_k] = 0$, $\mathbb{E}[\|\xi_k\|^2 \mid \theta_k] \le \sigma^2$. Since $\mathcal{L}^*$ is L_{lip}-smooth (A1),

$$\mathcal{L}^*(\theta_{k+1}) \le \mathcal{L}^*(\theta_k) - \eta\langle\nabla\mathcal{L}^*(\theta_k), \hat{G}_k\rangle + \frac{\eta^2 L_{\text{lip}}}{2}\|\hat{G}_k\|^2.$$

Taking expectation and using the decomposition of $\hat{G}_k$:

$$\mathbb{E}[\mathcal{L}^*(\theta_{k+1})] \le \mathbb{E}[\mathcal{L}^*(\theta_k)] - \eta\mathbb{E}[\|\nabla\mathcal{L}^*\|^2] + \eta\mathbb{E}[\langle\nabla\mathcal{L}^*, -b_k\rangle] + \frac{\eta^2 L_{\text{lip}}}{2}\mathbb{E}[\|\hat{G}_k\|^2].$$

Young's inequality: $\langle\nabla\mathcal{L}^*, -b_k\rangle \le \frac{1}{2}\|\nabla\mathcal{L}^*\|^2 + \frac{\mathcal{B}^2}{2}$, and $\|\hat{G}_k\|^2 \le 2\|\nabla\mathcal{L}^*\|^2 + 2\mathcal{B}^2 + 2\|\xi_k\|^2$. Substituting:

$$\mathbb{E}[\mathcal{L}^*(\theta_{k+1})] \le \mathbb{E}[\mathcal{L}^*(\theta_k)] - \frac{\eta}{2}\mathbb{E}[\|\nabla\mathcal{L}^*\|^2] + \frac{\eta\mathcal{B}^2}{2} + \eta^2 L_{\text{lip}}\big(\mathbb{E}[\|\nabla\mathcal{L}^*\|^2] + \mathcal{B}^2 + \sigma^2\big).$$

For $\eta \le 1/(4L_{\text{lip}})$, the coefficient of $\mathbb{E}[\|\nabla\mathcal{L}^*\|^2]$ is at most $-\eta/4$:

$$\mathbb{E}[\mathcal{L}^*(\theta_{k+1})] \le \mathbb{E}[\mathcal{L}^*(\theta_k)] - \frac{\eta}{4}\mathbb{E}[\|\nabla\mathcal{L}^*(\theta_k)\|^2] + \frac{\eta\mathcal{B}^2}{2} + \eta^2 L_{\text{lip}}(\mathcal{B}^2 + \sigma^2).$$

Summing over $k = 0, \dots, K-1$ and telescoping:

$$\frac{\eta}{4}\sum_{k=0}^{K-1}\mathbb{E}[\|\nabla\mathcal{L}^*(\theta_k)\|^2] \le \Delta_0 + K\frac{\eta\mathcal{B}^2}{2} + K\eta^2 L_{\text{lip}}(\mathcal{B}^2 + \sigma^2).$$

Dividing by $\eta K/4$ and using min ≤ average:

$$\min_{0\le k\le K-1} \mathbb{E}[\|\nabla\mathcal{L}^*(\theta_k)\|^2] \le \frac{4\Delta_0}{\eta K} + 2\mathcal{B}^2 + 4\eta L_{\text{lip}}(\mathcal{B}^2 + \sigma^2).$$

With $\eta = 1/\sqrt{K}$ and $K \ge 16L_{\text{lip}}^2$, the third term is $4L_{\text{lip}}(\mathcal{B}^2 + \sigma^2)/\sqrt{K}$, giving the stated constant. □

[1] Convention: $\hat{G}_k$ is a minibatch estimate of the *semi-gradient* $\tilde{\nabla}\mathcal{L}$, so the sampling noise ξ_k in A3 is the difference between $\tilde{\nabla}\mathcal{L}$ and its minibatch estimate. We do *not* assume a minibatch estimate of $\nabla\mathcal{L}^*$; the latter appears only in the analysis as a deterministic comparison point.

Remark 6 (Interpretation). *The rate $O(1/\sqrt{K})$ is optimal for nonconvex stochastic optimization with bounded gradients (Ghadimi & Lan, 2013), stated as a min-over-iterates bound. The coefficient 2 on $\mathcal{B}^2$ comes from the Young's-inequality step $\langle \nabla\mathcal{L}^*, -b_k\rangle \le \frac{1}{2}\|\nabla\mathcal{L}^*\|^2 + \frac{1}{2}\mathcal{B}^2$ combined with the bound on $\|\hat{G}_k\|^2$; it is a valid bound, not a tight constant. A sharper coefficient would require a specific noise model for b_k (e.g., assuming b_k is approximately aligned with $\nabla\mathcal{L}^*$), which we do not assume. The floor is therefore stated as an upper bound; the true floor under the STE gap may be smaller but is not provably improvable to $\mathcal{B}^2$ in the worst case. The constant C_1 depends only on Δ_0, L_{lip}, σ^2, and $\mathcal{B}^2$. The first three are T- and n-independent (smoothness of the averaged objective equals smoothness of a single term; variance is set by minibatch size). The $\mathcal{B}^2$ term depends on T only through $\sup_t \|m^t\|$, which is bounded by the message-head Lipschitz constant in practice.*

3.7 Number of Updates to Reach an ϵ-Optimum

Corollary 8. *Fix $\epsilon > 2\mathcal{B}^2$ and $\delta = \epsilon - 2\mathcal{B}^2 > 0$. Then $K = \mathcal{O}(C_1^2/\delta^2)$ updates suffice to reach $\min_{0\le k\le K-1} \mathbb{E}[\|\nabla\mathcal{L}^*(\theta_k)\|^2] \le \epsilon$.*

Proof. Set $C_1/\sqrt{K} \le \delta$ and solve for K. □

3.8 Credit Assignment at Bounded Graph Depth

The central claim. The contribution of this section is not any single ingredient. STE (Bengio et al., 2013), linear projections, and two-pass backward algorithms each exist in isolation. What is new is that the *linear-in-pos* token construction makes the STE bias directionally zero along the message gradient (Lemma 1), which in turn makes the credit-assignment horizon $T - t$ achievable at graph depth $\Theta(1)$ per step (Proposition 4), with a trade-off quantitatively controlled by c_m (Lemma 5). Removing any one of the three collapses the construction: without linearity, the STE bias is directional and the horizon is not exactly $T - t$; without a bounded STE bias, the horizon is not preserved under iteration; without $c_m/(n-1)$, the contractivity coefficient grows with n and the $T - t$ horizon collapses to a few steps (Proposition 4).

Summary. TINAS achieves a credit-assignment horizon of $T - t$ steps at autograd graph depth $\Theta(1)$ per step. This is a structural property of the algorithm, not a claim about any specific alternative mechanism.

Graph depth per step. At step t, the autograd graph is built only for the local forward pass `extension->forward`$(r[t], m[t-1])$. Both inputs are detached from Pass 1, so no gradient edge crosses from t to $t-1$. Graph depth equals the depth of the extension transformer, independent of T and n.

Credit horizon. By Proposition 4, g^t depends on $b^{t'}$ for all $t' \in [t, T]$, with weight $\le (c_m L_{\text{net}})^{t'-t}$. The credit horizon is $T - t$, with geometric decay.

The price. The bounded STE bias of Lemma 5, controlled by c_m, message magnitude, and B_2. Corollary 6 shows that this is the only price: as $\mathcal{B} \to 0$, the TINAS gradient approaches the exact BPTT gradient.

Memory comparison. Compare with methods that propagate the gradient through a full T-step computational graph (e.g., DIAL). Let A be per-step activation memory and $S = n(d_{\text{msg}} + d_r)$ per-step snapshot memory. For a transformer with L layers and hidden dimension d: $A = \Omega(ndL)$, $S = \mathcal{O}(nd)$, so $A/S = \Omega(L)$.

	Full-graph	**TINAS**
Graph depth per step	$\Theta(T)$	$\Theta(1)$
Activation memory	$\Theta(T \cdot A)$	$\Theta(T \cdot S + A)$
Credit horizon	$T - t$	$T - t$ (geometric decay)
Total asymptotic memory	$\Theta(T \cdot A + \lvert\theta\rvert)$	$\Theta(T \cdot S + A + \lvert\theta\rvert)$
Ratio (TINAS / full-graph)	1	$\Theta(S/A + 1/T)$

Table 2: Comparison with methods using a full T-step graph. A = per-step activation memory; $S = n(d_{\text{msg}} + d_r)$ = per-step snapshot memory.

For $T \gg 1$ and $T \cdot S \gg A$, the ratio is $\Theta(S/A) = \mathcal{O}(1/L)$—independent of n and T.

4 Algorithm: Two-Pass Backward

Pass 1 — forward without gradients.

```
m[0] = 0
with NoGradGuard:
    for t = 1, ..., T-1:
        r[t]  = model->forward_r_t(obs[t]).r_t().clone().detach()
        m[t]  = extension->forward(r[t], m[t-1]).new_messages().clone().detach()
model->reset_memory()
```

$r[t]$ and $m[t]$ are recorded as detached tensors. The backbone is frozen; only the extension is updated.

Pass 2 — backward with injected gradients.

```
g[T+1][i][j] = 0 for all i, j
G = 0

for t = T, T-1, ..., 1:
    out = extension->forward(r[t], m[t-1])

    for i = 1, ..., n:
        L_msg = 0
        for j != i:
            L_msg += (g[t+1][i][j] . m[t][i->j]).sum()
        L_msg /= (n - 1)

        L_i^t = L^{i,t}_PPO + c_V L^{i,t}_V - c_H H^{i,t} + c_m L_msg
        grad_i = autograd.grad(L_i^t, params)

        for k != i:
            g[t][k][i] = grad_i.pos[k]      # incoming convention

        grad_i = clip_norm_1(grad_i)
        G += grad_i

G /= (T * n)
theta <- AdamW(G, lr=eta, wd=1e-4)
```

Indexing convention. $g[t][i][j]$ stores $g^t_{j \to i} = \partial L^t_i / \partial \text{pos}[j]$, i.e., the *incoming* convention (receiver first, sender second). Storing in the outgoing convention would pair the wrong gradient with $m^t_{i \to j}$.

Two clarifications.

1. **No re-attach.** Pass 2 uses $\partial L_i^t / \partial \text{pos}$ directly; by Lemma 1, this equals the message-side gradient.
2. **Pass 2 never touches `PlayerPolicyNet`.** $r[t]$ is detached, so the sliding-window buffer is irrelevant during Pass 2.

Cost. Pass 1 is $O(T)$ forwards. Pass 2 is $O(T)$ forwards plus n backward calls per step, which can be combined into one at the cost of losing per-agent clipping. Both are $O(T)$ asymptotically.

5 Practical Guarantees under AdamW + Gradient Clipping

The analysis is stated for SGD. The implementation uses AdamW ($\eta = 10^{-3}$, $\lambda = 10^{-4}$) with clipping at norm 1. We state which pieces carry over and cite the analysis rather than re-derive.

Clip 1 enforces A2. The clipped update satisfies $\|\hat{G}_k^{\text{clip}}\| \le 1$, so $\|\theta_{k+1} - \theta_k\| \le \eta$. Clipping bounds the *update*, not the raw gradient.

AdamW inherits a comparable rate under standard conditions. We do not analyze AdamW here. Under L_{lip}-smoothness, $\beta_2 < 1$, bounded stochastic-gradient variance, and bounded second-moment estimates, Adam and AdamW converge to a stationary point at rate $O(\log K/\sqrt{K})$ (Kingma & Ba, 2015; Chen et al., 2019; **?**). These conditions are compatible with A1–A3 (which supply smoothness and bounded variance) but stronger in that Adam's analysis additionally requires bounded second-moment estimates. Under those conditions the $\mathcal{B}^2$ floor enters the bound unchanged; the constant is adjusted for the adaptive per-coordinate step.

6 Characterization of the Stationary Point

At any stationary point θ^* of $\mathcal{L}^*$, $\nabla_\theta \mathcal{L}^*(\theta^*) = 0$ reads

$$\frac{1}{Tn} \sum_{t,i} \left[\nabla_\theta \left(L_{\text{PPO}}^{i,t} + c_V L_V^{i,t} - c_H H^{i,t} \right) + \frac{c_m}{n-1} \sum_{j \neq i} \left(g_{i \to j}^{t+1} \right)^\top \nabla_\theta m_{i \to j}^t \right] \; + \; \text{STE correction} \; = \; 0,$$

where the STE correction is $c_m \sum_j (\partial g_{i \to j}^{t+1} / \partial \theta)^\top m_{i \to j}^t$, bounded by $\mathcal{B}$.

This is a balance between the PPO signal, the message-injection signal, and the STE correction. At equilibrium (up to $\mathcal{B}$), no unilateral perturbation of any agent's messages can improve that agent's message-loss contribution while keeping all other parameters and messages fixed. This is a weak analogue of Nash equilibrium for the message-passing game.

7 Related Work

Communication in MARL. DIAL (Foerster et al., 2016) is the closest predecessor: agents send continuous messages over a differentiable channel, and gradients flow through the receiver's loss into the sender during training. DIAL is Q-learning-based and propagates gradients through a full T-step graph; TINAS is PPO-based and operates at graph depth $\Theta(1)$ per step. The two are complementary in the outer-RL dimension and orthogonal in graph structure; Section 3.8 addresses the latter. CommNet (Sukhbaatar et al., 2016) uses a mean-pooled channel; TarMAC (Das et al., 2019) uses attention-weighted messages with end-to-end backprop; IC3Net (Singh et al., 2019) learns discrete gates. TINAS is complementary architecturally and orthogonal computationally.

Straight-through estimators. The STE (Bengio et al., 2013) is typically used to backpropagate through non-differentiable operations. TINAS uses it to avoid storing the message graph across T timesteps. Lemma 5 bounds the bias explicitly, which is uncommon in STE applications where the bias is usually treated as part of the modeling assumption.

Transformers for MARL. Multi-head self-attention for inter-agent communication is well established (Vaswani et al., 2017; Wang et al., 2018). TINAS uses a pre-LN transformer block (Xiong et al., 2020) with a dedicated positional encoding for sender identity. The linear-in-pos construction enables Lemma 1.

Parameter-sharing MARL. Sharing parameters is standard for scaling MARL (Foerster et al., 2016; Rashid et al., 2018). TINAS assumes a single shared extension network with per-agent positional embeddings.

8 Scope

The following are explicitly *not* claimed.

- That TINAS learns *semantically meaningful* messages. The objective trains messages to reduce the receiver's future loss; message semantics are not analyzed.
- That the STE bias $\mathcal{B}$ is negligible at the operating c_m. Lemma 5 bounds it in terms of c_m, $\sup_t \|m^t\|$, and B_2.
- That the contractivity condition $c_m L_{\text{net}} < 1$ holds throughout training. A c_m-schedule or normalization on the message head would enforce it more robustly and is left for future work.
- That the empirical evaluation of Section 10 has been completed. The current manuscript reports the theoretical framework, algorithmic design, and implementation; empirical results are planned.

The convergence statement of Theorem 7 is local under A1, A3, A5, and does not assert any correspondence between stationary points of $\mathcal{L}^*$ and optimal policy performance. Stationary points of $\mathcal{L}^*$ are the natural reference because a fully-differentiable method in the same architectural setup would optimize exactly this objective (Corollary 6).

9 Implementation

We instantiate TINAS on a behavioral-cloning backbone trained in the *StrikeForce* 2D testbed (Fouladi, 2026b), using the FAP/Transformer-aggregator architecture of (Fouladi, 2026a). The backbone is frozen; the extension is trained by TINAS. Code and checkpoints at `https://github.com/kasrafouladi/StrikeForce-No-BPTT-via-FAP`.

9.1 Architecture

Shared backbone. `PlayerPolicyNet` consists of an AFC backbone (`AFCBackboneSparse`) with two sparse-attention stages, a transformer policy block (`tf_policy`, $d_{\text{model}} = 300$, 10 heads), and a sliding window of $W = 31$ frames. Output: policy logits over 7 actions and $r_t \in \mathbb{R}^{300}$. Pretrained, frozen.

Communication extension. `CommExtensionNet`: per-agent sequence of length $n + 1$, [r_i^t, tok_value, $m_{j_1 \to i}, \ldots, m_{j_{n-1} \to i}$]. The r_t slot carries rt_pe; the value token has no PE; incoming messages are tagged with sender_pe[j], concatenated with $d^t[j \to i]$, and projected by `msg_proj` to $d_r = 300$.

Heads. `policy_head` (on r_t), `value_head` (on value token), `msg_out_head` (on each message slot).

Token construction per Lemma 1.

```
msg_with_pe = incoming + sender_pe[other_idx[i]]
msg_feat    = cat({msg_with_pe, dist_in}, -1)
msg_emb     = msg_proj(msg_feat)
```

`incoming` detached; `sender_pe` attached. Linear `msg_proj` on the sum makes the pos-gradient equal to the message-gradient.

Gather indices. `other_idx` is a buffer $[n, n-1]$ with `other_idx[i]`$=[j \neq i]$.

9.2 Per-Step Composition

Each optimization step sums three gradient contributions (PPO, value, message), clips each at norm 1, accumulates, divides by Tn, and steps AdamW.

9.3 Hyperparameters

Hyperparameter	Value
c_m / c_V / c_H	0.1 / 0.5 / 0.01
AdamW η / λ	10^{-3} (constant) / 10^{-4}
d_{msg} / d_{dist}	300 / 2
n / ext. heads / clip norm	5 / 10 / 1

Table 3: Configuration; values are not claimed to be optimal.

10 Planned Evaluation

We plan to compare TINAS against the frozen backbone alone on StrikeForce, using paired episodes with identical seeds and action masks. Metrics follow the benchmark (Fouladi, 2026b): kill rate, kills per unit stamina, damage rate, and composite M_{HP}, M_{stam}, M_{dmg}.

Ablations: (i) `sender_pe` replaced with fixed identity; (ii) value token removed; (iii) distance concatenation removed; (iv) shared vs. independent extensions. These isolate the future-gradient injection mechanism from the architectural choices of the extension.

References

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Léon Bottou, Frank E. Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311, 2018.

Xiangyi Chen, Sijia Liu, Ruoyu Sun, and Mingyi Hong. On the convergence of a class of Adam-type algorithms for non-convex optimization. In *International Conference on Learning Representations*, 2019.

Abhishek Das, Théophile Gervet, Joshua Romoff, Dhruv Batra, Devi Parikh, Mike Rabbat, and Joelle Pineau. TarMAC: Targeted multi-agent communication. In *International Conference on Machine Learning*, 2019.

Jakob Foerster, Ioannis Alexandros Assael, Nando de Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*, 2016.

Kasra Fouladi. No back-propagation through time via future action prediction. ResearchGate, 2026a. URL https://doi.org/10.13140/RG.2.2.18971.43049.

Kasra Fouladi. Strikeforce: A 2d partially observable battle-arena testbed for imitation learning and multi-agent research. ResearchGate, 2026b. URL https://doi.org/10.13140/RG.2.2.24446.96326.

Saeed Ghadimi and Guanghui Lan. Stochastic first- and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 23(4):2341–2368, 2013.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2015.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder de Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International Conference on Machine Learning*, 2018.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. In *arXiv preprint arXiv:1707.06347*, 2017.

Amanpreet Singh, Tushar Jain, and Suresh Veeraraghavan. Individualized controlled continuous communication model for multiagent cooperative and competitive tasks. In *International Conference on Learning Representations*, 2019.

Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. Learning multiagent communication with backpropagation. In *Advances in Neural Information Processing Systems*, 2016.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tie-Yan Liu. On layer normalization in the transformer architecture. In *International Conference on Machine Learning*, 2020.