DirCast: Live Esports Directing and Commentary with a Single Streaming Video-Language Model

Zixian Gao[1,2], **Lixin Xiu**[2], **Penghao Luo**[3], **Xiangcheng Cao**[1], **Ziqi Cai**[1,4,5], **Zhen Li**[1,2], **Erwin Wu**[5], **Kaipeng Zhang**[1]

[1]Alaya Lab [2]The University of Tokyo [3]Georgia Institute of Technology
[4]Peking University [5]Institute of Science Tokyo

A live esports broadcast needs a director who chooses which camera goes on air and a caster who talks over that choice, every second and without knowing what comes next. Streaming video language models have learned to commentate, but only over a broadcast that a human director has already produced. We study live broadcast production, in which a model receives the raw camera of every player and must both direct and commentate from the past alone. We present DirCast, a single streaming video language model that does both. Every second it scans all cameras at low resolution, decides with a hold-or-switch director whether and where to cut, and reads the chosen camera at high resolution before it speaks. Its context keeps what went on air and what was said and lets the other views expire, which bounds the cost of a stream of any length. We build a dataset of 27 professional Dota 2 matches with a camera for every hero, the human director's choices and the casters' commentary. DirCast directs better than rule-based and zero-shot directors including GPT-5, and its commentary wins 29.2 percent of comparisons against GPT-4o under the Proact-VL protocol, against 19.4 percent for the strongest of eleven streaming commentators that watch the human broadcast.

Project Page: https://zixiangao.github.io/DirCast/
Code: https://github.com/ZixianGao/DirCast-code
Correspondence: Kaipeng Zhang
Date: October 24, 2026

ALAYA LAB

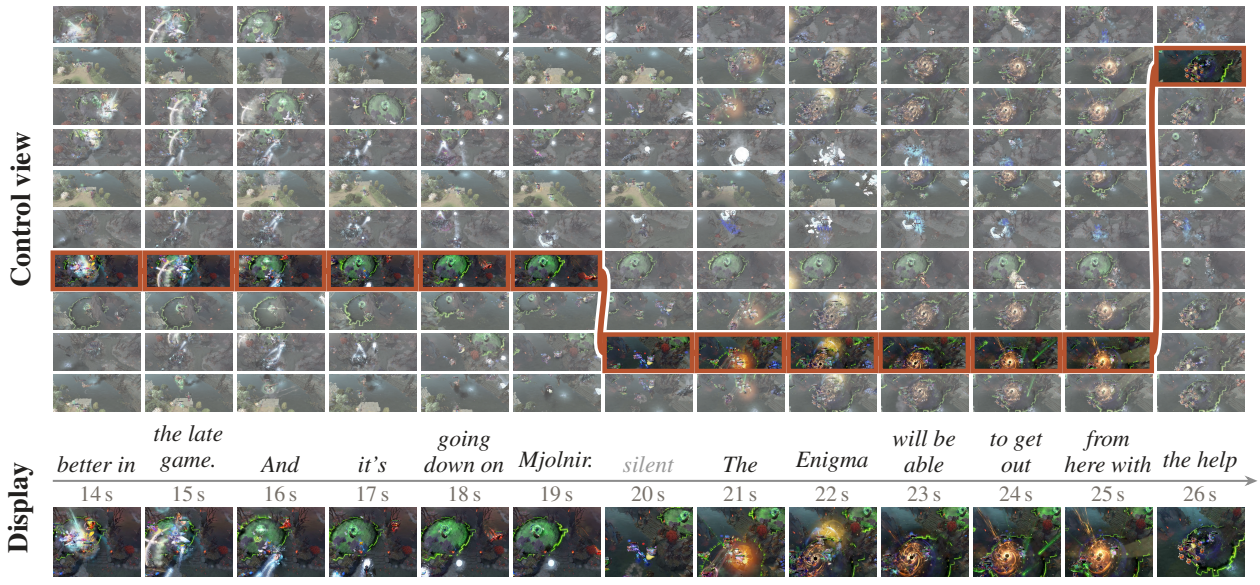


Figure 1 DirCast directs and commentates a live Dota 2 match. Top: the ten hero cameras over 13 seconds, with the on-air camera framed. Bottom: the program that goes on air and the words DirCast speaks over it.

1 Introduction

A live esports broadcast is produced by two people working in tight coordination. The director watches a wall of camera feeds, one per player, and decides every second which feed goes on air. The caster watches what the director puts on air and talks over it, naming the heroes, calling the fights, and filling the quiet moments with analysis. Neither job can wait for the future. When a fight breaks out on the far side of the map, the director has a few seconds to cut to it, and the caster has to describe it while it is still happening. This is why professional broadcasts need trained crews, and why the long tail of matches, from regional leagues to amateur games, is streamed with a fixed camera and no commentary.

Recent video language models [2, 18, 37] have started to take over the second job. Streaming commentators such as LiveCC [8], StreamingVLM [42] and Proact-VL [43] read a video one frame at a time and decide when to speak and what to say [7, 29, 38, 39, 44]. They all assume, however, that the first job has already been done. Their input is a finished broadcast in which a human director has chosen every shot. The model inherits the director's judgment for free, and the hardest part of live production, deciding what the audience should see, is left outside the problem. A system that is given only the raw feeds of a match cannot use these models at all.

We study the full problem. Given the ten hero cameras of a live Dota 2 match and the game events as they happen, a model has to choose the on-air feed every second and produce live commentary over its own choice, using only the past. We call this task live broadcast production. It couples two decisions that are usually studied apart [5, 6, 30, 45]. The director must track ten simultaneous scenes and recognize which one is about to matter [4, 23]. The commentator must describe a picture that changes whenever the director cuts, and its words are only good if the director put the right action on screen. A model that solves the task produces a complete broadcast from raw input.

The coupling makes the task hard in ways that neither half shows alone. Ten feeds at one frame per second produce more visual tokens than a streaming model can keep in context for long [9, 28, 32, 46]. The decisions compound over time, because a director that misses a fight shows the commentator the wrong scene, and every later second is conditioned on that early mistake. Training on human broadcasts does not prepare the model for this, since at training time it only ever sees the program a human chose, the classic exposure bias [3, 31]. Human directors also hold a shot most of the time and cut rarely, so a model trained to imitate them learns to never cut at all. And a streaming model that is allowed to stay silent tends to fall into silence and never fully recover.

We present DirCast, a single causal video language model that directs and commentates at once. Every second, the model reads a short chunk that contains the new game events and the ten feeds at low resolution, each paired with a token that encodes the hero's game state. A hold-or-switch director head then decides whether to keep the current shot or to cut, and a pointer over the ten feeds chooses where to cut to. The chosen feed is appended to the stream at high resolution, and the commentary is generated after it, so the commentator always speaks over the shot the director chose. Low-resolution feed tokens leave the context after a few seconds, while the on-air views and the spoken words stay. The memory of the model is the program it has produced, which keeps the cost per second bounded and lets DirCast run in real time on a single GPU.

Three further choices make the model robust when it runs on its own. We roll out perturbed programs during training, in which the on-air feed drifts away from the human choice for a few seconds while the targets stay human, so the director learns to recover from its own mistakes and the commentator learns to speak over any shot. We ground every kill in the event stream to the feed of the hero who made it, and we accept a cut to a fight participant as a correct directing decision, which teaches the director that action is worth showing even when the human chose otherwise. At inference, the thresholds for cutting and for speaking follow the running rates of the model's own decisions toward the rates of human crews, which keeps the program lively and the commentary present without any tuning on test data.

To study the task we build a dataset from 27 professional Dota 2 matches. We render one camera per hero from the match replays, align the renders with the official broadcast, and label every second with the feed the human director put on air and the words the casters spoke. We evaluate commentary with the protocol

of Proact-VL [43] and direct comparison against eleven streaming commentators, and we evaluate directing by agreement with the human director and by coverage of the kills in the match. DirCast wins 29.21 percent of its comparisons against GPT-4o, while the strongest streaming commentator wins 19.38 percent, even though every baseline is given the finished human broadcast and DirCast directs its own. It starts speaking 0.30 seconds after the human casters on average. As a director, it agrees with the human crew more often than every rule-based and zero-shot director, it covers 89.1 percent of kills, and it cuts at a rate close to the human crew. The best zero-shot vision language model agrees with the human crew only two points more often than a static director that never cuts, and GPT-5 agrees less often. Trained as one model, DirCast is also better at both roles than two models trained separately, with half the memory.

Our contributions are as follows.

- We introduce live broadcast production, directing and commentating a live match from raw camera feeds, with a dataset of 27 professional Dota 2 matches.
- We propose DirCast, a single streaming video language model that decides what goes on air with a hold-or-switch director and comments on the shot it chooses, in real time.
- DirCast outperforms eleven streaming commentators while directing its own broadcast, and directs better than rule-based and zero-shot directors, including GPT-5.

2 Related Work

Streaming video language models. Video language models such as Video-LLaVA [20], LLaVA-OneVision [18], InternVL3.5 [37], Qwen2.5-VL and Qwen3-VL [2] process prerecorded clips, whereas streaming models must respond as video arrives. VideoLLM-online [7] learns when to remain silent, while subsequent work uses separate decision modules [29, 35], interleaved dialogue [38, 39] or learned response timing [44]. Long-term context is managed through memory banks [12, 32], compression [9, 28, 46], KV-cache retrieval [9], recent-window attention [40], token eviction [47] and compact frame representations [19, 41]. StreamingBench [21] and OVO-Bench [26] evaluate online understanding. For live commentary, LiveCC [8] learns from large-scale speech supervision, StreamingVLM [42] supports hours of commentary with bounded memory, and Proact-VL [43] introduces proactive gaming commentary and an evaluation protocol we adopt. These methods assume a preselected video stream. DirCast instead produces that stream through causal directing and maintains role-aware memory of aired views and spoken commentary.

Commentary for sports and games. Automated commentary spans chess [15], online video [22] and soccer, where SoccerNet [10] supports event captioning [24], knowledge-grounded commentary [27] and timeline-aligned narration [30]. Games provide structured records for commentary generation, including racing with video and telemetry [14] and esports datasets pairing League of Legends or Dota 2 gameplay with professional commentary [34, 45, 48]. These approaches rely on a single produced broadcast or game data without video. Our dataset preserves all available camera views, enabling joint training and evaluation of shot selection and commentary.

Automatic directing and view selection. Automatic cinematography includes editing multi-camera social footage [1], learning dialogue editing conventions [17] and assembling videos from text [36]. Sports systems learn camera control [4] and view selection based on visual importance [5, 6]. Related work selects viewpoints in panoramic video [13, 33] or informative views in multi-camera instructional recordings [11, 23]. Unlike offline view selection, DirCast directs causally at each second and couples each decision with commentary. Its coarse-to-fine processing resembles glimpse-based attention [25]. Since directing decisions alter subsequent observations, the model also faces exposure bias, studied through scheduled sampling [3], professor forcing [16] and expert-corrected imitation learning [31]. Our perturbed program rollouts similarly expose the director to states induced by its own errors.

3 Dataset

Live broadcast production needs data that no existing video corpus provides. A broadcast shows one camera at a time, and the cameras the director did not choose are lost. To learn directing, a model has to see every camera the director could have chosen, together with the one the director actually put on air and the words the casters spoke over it.

Esports make this possible. A professional Dota 2 match is fully described by its replay file, so the game can be played back and filmed from any point of view after the fact. We collect 27 recent professional matches with their official broadcasts. For every match we render ten cameras from the replay with the game's own spectator client, one following each hero. We then align the rendered timeline with the broadcast and label every second with three complementary signals. The first is the on-air camera, the hero feed that best matches what the broadcast shows at that second. The second is the state of the broadcast, which separates live gameplay from replays, overlays and breaks, so that only live seconds count as directing decisions. The third is the commentary, transcribed from the broadcast audio with word timestamps and cleaned following the Proact-VL polishing pipeline, which corrects misheard hero and player names while keeping the timing of every word. From the replay we further extract the state of every hero at every second, such as health, nearby enemies and recent damage, and the stream of game events, such as kills and destroyed buildings, each placed at the second it happens.

We cut the aligned matches into clips of 36 seconds at one frame per second, the setting used by streaming commentary benchmarks. The split is by match, with 20 matches for training, 3 for calibration and 4 for testing, so that no team, player or game of the test set is seen in training. The training set contains 2,096 clips and 21 hours of play, and the test set contains 496 clips and 5 hours. Human directors cut 7.9 times per minute on average and hold most shots for several seconds. The casters speak in about three quarters of all seconds. These two numbers describe the rhythm a model has to reproduce, and we return to them when we discuss inference.

4 Method

Every second, DirCast receives the ten camera frames and the new game events, decides which camera goes on air, and decides whether and what to say. It may only use the past. We build the model around one observation. A director and a commentator are not two separate systems that happen to run at the same time. The director decides what the audience sees, the commentator talks about what the audience sees, and both depend on what has been shown and said so far. We therefore use a single causal video language model for both roles, in which the commentary is generated after, and conditioned on, the directing decision. Figure 2 gives an overview.

4.1 One stream for directing and commentary

We start from a pretrained video language model and feed it the match as a single token stream that grows by one chunk per second. The chunk for second t has a fixed layout. It opens with the game clock and the events of that second. It then lists the ten cameras, each as a low-resolution view v_t^k preceded by a state token s_t^k that embeds the game state of hero k. A director token $\langle \mathrm{D} \rangle$ follows. After the directing decision, the chosen camera c_t is appended again at high resolution, followed by a commentary token $\langle \mathrm{C} \rangle$ and the words w_t spoken in that second.

The order of the chunk follows the order in which a human crew works. The director scans all cameras at a glance, picks one, and the commentator then looks closely at the picture that went on air. Each step reads everything before it, so the director sees the full history of the broadcast and the commentator sees the director's choice. The two resolutions keep the stream affordable. Ten cameras at full resolution would produce far more visual tokens than a streaming model can process every second, and most of them show nothing worth describing. A low-resolution view is enough to tell where the action is, while the commentator needs a sharper view to name heroes, items and abilities. DirCast therefore reads only the camera that goes on air at high resolution.

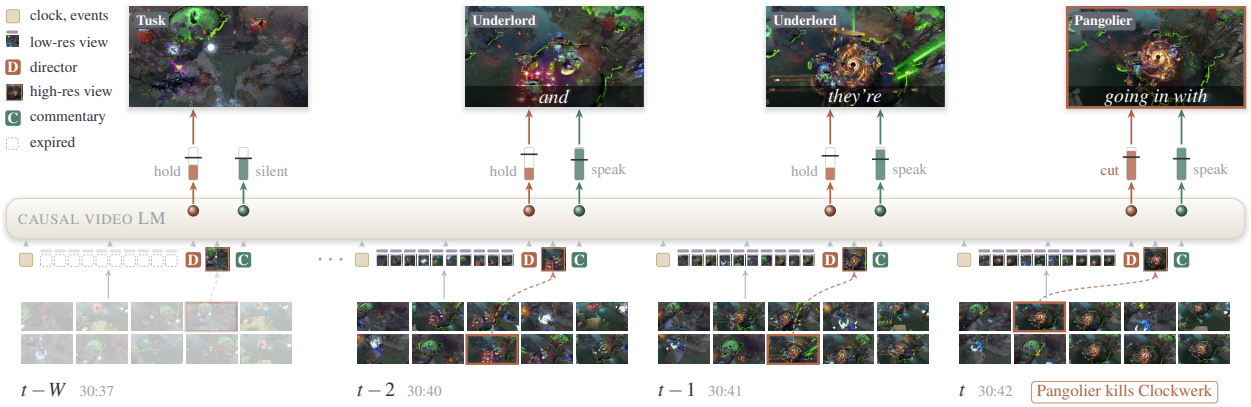


Figure 2 Framework. Each second a low-res ViT encodes the ten feeds with their state, the director holds or cuts, the chosen feed is read at high resolution, and the commentator speaks. Low-res views expire while the program stays.

4.2 Hold or switch

Most of the time a good director does nothing. Shots are held for several seconds, and a cut is a deliberate event that happens when something more important appears elsewhere. A classifier over ten cameras does not capture this structure. It must relearn at every second that the current camera is usually the right one, and small changes in its scores produce flickering cuts.

We instead factor the decision into whether to cut and where to cut to. Let h_{D} be the hidden state of the director token and let $\bar{h}_k$ be the mean hidden state of the tokens of camera k. A cut gate

$$g_t = \sigma\big(f([h_{\mathrm{D}}\,,\,\bar{h}_{c_{t-1}}]) + \psi(s_t)\big) \tag{1}$$

estimates the probability of leaving the camera that is on air, given the director state and the current shot. A pointer

$$\pi_t(k) \propto \exp\big\langle q(h_{\mathrm{D}})\,,\,\kappa(\bar{h}_k) + e_k + \phi(s_t^k)\big\rangle, \quad k \neq c_{t-1}, \tag{2}$$

scores the other cameras by comparing the director state with each camera's content and game state, where e_k is a learned embedding of the player slot. The probability of each camera is then

$$P(c_t = k) = (1 - g_t)\,\mathbf{1}[k = c_{t-1}] + g_t\,\pi_t(k). \tag{3}$$

Holding is the default, and switching requires the gate to open. The pointer treats all cameras with the same function, so it ranks cameras by what they show instead of by their position in the input.

4.3 The program is the memory

A streaming model must forget, or its context grows without bound. The question is what to keep. A human crew does not remember what every camera showed a minute ago. It remembers the broadcast, the shots that actually went on air and what was said over them. DirCast keeps this. The low-resolution views of the ten cameras stay for only W seconds and then leave the context, while the high-resolution on-air views, the events and the words stay. During training this rule is an attention mask, and during streaming the expired entries are removed from the key-value cache. The model's memory is the program it has produced. On streams longer than a training clip, every second also leaves the cache once it is older than a clip, so the cost per second stays constant over a match of any length.

4.4 Grounding the game

Video alone is a weak signal for directing. A fight that is about to start looks like two groups of heroes walking, and a kill on the far side of the map is a small change in one of ten thumbnails. The game itself knows more. The state token of each camera gives the model the hero's health, recent damage and the enemies around it, and the event text states what happened, such as one hero killing another. We tie each event to the camera that shows it, so that the kill of a hero raises a flag on the camera of the hero who made it. The model thus reads the game in the same terms as the cameras it has to choose from.

4.5 Learning to direct from human broadcasts

DirCast is trained on human broadcasts with three objectives. A language modeling loss teaches the commentary, a binary loss on the commentary token teaches when to speak, and a directing loss teaches the gate and the pointer. The gate learns from the human cuts, and the pointer learns which camera the human chose at every cut, where a camera whose hero is visible in the broadcast shot counts as a correct answer as well.

Imitating human broadcasts directly leaves two gaps. The first gap is between training and deployment. During training the model always sees the program the human chose. At test time it sees its own program, and a single wrong cut moves it to a state it has never seen. We close this gap with on-policy program rollouts. During training, the on-air camera in the input drifts away from the human choice for a few seconds at a time, while the targets stay human. The director learns to recover from a wrong shot by cutting back, and the commentator learns to describe whatever is on air. The second gap is in the targets. A human director makes one choice among several good ones. When a kill happens on a camera that is not on air, cutting to it is a correct directing decision even if the human stayed elsewhere. We add an event-anchored target that, in the seconds around a kill, also accepts the cameras of the killer and the victim, and asks the gate to open when neither is on air. The model learns that action is worth showing, which is the part of directing that the human labels express least consistently.

4.6 Streaming decisions at a human rhythm

At inference, the model cuts when the gate exceeds a threshold and speaks when the probability of the commentary token exceeds a threshold. Fixed thresholds fail in streaming. A model that stays silent for a few seconds conditions on its own silence and becomes more likely to stay silent, and a director that has not cut for a while drifts the same way. We let both thresholds follow the model's own behavior,

$$\tau_t = \tau_0 + \gamma\,(\bar{r}_t - r^*), \tag{4}$$

where $\bar{r}_t$ is a running average of the model's recent decisions and r^* is the rate of human crews measured on the training set. When the model has been quieter than human casters, the threshold to speak drops, and when it has been cutting more often than human directors, the threshold to cut rises. The running rate of speech starts at zero, since the model has not spoken when it joins a stream. Once the model starts an utterance, it ends it only when its end probability is high, so that each second carries a complete phrase rather than a fragment. These rules carry the rhythm of a human broadcast into the model without any training signal and keep the broadcast lively for as long as it runs.

5 Experiments

5.1 Setup

Protocol. All results are on the test split of four unseen matches, 496 clips with 485 commented and 11,283 strict LIVE seconds. Every model runs free through every second of each clip and conditions only on its own past decisions.

Directing. We measure how often a model agrees with the human director (Top-1, Tol-top1 with one second of slack, and Any-visible, which accepts any feed whose hero is visible on air). We also report the precision and F1 of correct cuts, and the coverage of kills and fights on air. We compare with three rules and eight zero-shot VLMs. The random rule cuts to a random feed at the human cutting rate, and the static rule keeps one feed throughout. The action rule scores every feed by the game state of its hero, nearby enemies, recent damage, deaths and crowding, and cuts to the highest-scoring feed when it leads and the current shot has been held at least three seconds. The zero-shot models [2, 18, 37] include proprietary GPT-5 and Kimi-K2.6 and see the same feeds, heroes and events as DirCast.

Commentary. We follow the Proact-VL evaluation protocol [43] and run its publicly released scripts. CC is the win rate against offline GPT-4o commentary, F1 and TimeDiff measure whether and when the model

Table 1 Test-split directing. A cut is correct if within two seconds of a human cut and to the same feed. Fight coverage measures seconds with a fighting hero on air. Cuts/min should approach the human rate of 7.9. DirCast averages three seeds. Best bold, second underlined.

	Feed agreement			Cuts			Coverage	
Director	Top-1 ↑	Tol. ↑	Any-vis. ↑	P ↑	F1 ↑	/min	Kill ↑	Fight ↑
Rules								
Random (human rate)	10.6	12.9	20.1	6.4	5.8	6.9	20.9	19.1
Static	17.1	20.4	23.0	0.0	0.0	0.0	20.9	19.2
Action rule	<u>25.4</u>	<u>30.3</u>	<u>36.1</u>	<u>13.9</u>	11.5	5.8	78.5	**62.4**
Zero-shot VLMs								
Qwen3-VL-8B	17.6	21.4	28.6	8.9	12.7	20.6	74.6	32.6
Qwen3-VL-32B	11.2	13.9	21.7	8.6	7.4	6.3	67.8	30.6
InternVL3.5-8B	19.1	22.9	28.9	6.2	10.8	56.3	72.9	24.1
InternVL3.5-38B	19.0	22.8	27.7	6.6	10.0	25.3	81.9	28.1
LLaVA-OV-7B	9.2	12.0	18.6	6.5	9.2	20.5	41.2	18.8
Mistral-Small-3.2-24B	13.6	17.0	24.2	6.1	9.3	26.7	59.3	23.3
Kimi-K2.6	16.7	20.4	28.4	8.2	<u>13.7</u>	42.4	81.4	30.1
GPT-5	13.5	16.7	24.8	6.7	11.6	51.7	<u>85.3</u>	33.0
DirCast (ours)	**27.0**	**32.0**	**41.8**	**15.8**	**14.1**	6.7	**89.1**	<u>41.1</u>
Human director	100	100	100	100	100	7.9	50.3	37.4

chooses to speak, LiveU and FinalQ rate the live delivery and the quality of the final transcript, and PAUC summarizes commentary quality over the whole stream. GPT-5.1, the official judge, is not available to us, so all models, including ours, are judged by GPT-5 instead. We compare with eleven streaming commentators [8, 29, 38, 39, 42–44]. Nine run zero-shot with their released weights, and LiveCC-7B and Proact-VL are also fine-tuned on our training split. Each of them watches the finished human broadcast rather than choosing its own shots from the feeds.

Implementation. DirCast fine-tunes Qwen3-VL-8B [2] with LoRA for three epochs on eight H100 GPUs, using the language modeling, speaking and directing losses of Section 4 together with perturbed program rollouts. The model reads the ten feeds at 128 pixels and the on-air feed at 288 pixels, and low-resolution views expire after $W = 8$ seconds. Inference runs on a single GPU.

5.2 Main Results

Directing. Table 1 compares DirCast with rule-based and zero-shot directors. DirCast is the best director on every measure of agreement with the human crew. It reaches a Top-1 of 26.95 and an Any-visible of 41.81, against 25.43 and 36.11 for the action rule, the strongest baseline, and it has the highest precision and F1 of correct cuts. It covers 89.1 percent of kills, more than any baseline and more than the human director, who often stays with a shot when a kill happens elsewhere. Zero-shot vision language models do not direct well, whatever their size. None of the eight reaches the action rule on agreement, scale does not help within a family, and GPT-5 reaches a Top-1 of only 13.53. They also lack a sense of rhythm. Most of them cut 20 to 56 times per minute, against 7.9 for the human crew, so that most of their cuts are wrong and their cut precision stays between 6 and 9 percent. DirCast cuts 6.7 times per minute. The action rule follows fights by construction and has the highest fight coverage at 62.4. DirCast reaches 41.1, ahead of every zero-shot model and of the human director at 37.4, even though it is trained to imitate that director.

Commentary. Table 2 compares DirCast with nine zero-shot streaming commentators and LiveCC fine-tuned on our data. DirCast wins 29.21 percent of its comparisons against GPT-4o, ten points above the strongest baseline, LiveCC-7B-Base at 19.38, and each seed beats LiveCC-7B-Base and Proact-VL with $p < 0.001$ under a paired bootstrap over clips. The lead holds although every baseline watches the human broadcast while DirCast comments on its own program. DirCast also starts speaking closest to the human casters, with a

Table 2 Streaming commentary on the test split under the Proact-VL protocol, judged by GPT-5. CC is the win rate against GPT-4o. †Fine-tuned on our training split. DirCast is the mean of three seeds.

					LiveU			FinalQ			
Method	Venue	CC ↑	F1 ↑	TimeDiff ↓	Time	Rate	TextU	Fidelity	Contin.	Subst.	PAUC ↑
Zero-shot, human program											
LiveCC-7B-Base [8]	CVPR'25	19.38	64.61	8.32	5.11	4.99	4.53	4.56	3.17	2.36	0.008
LiveCC-7B-Instruct [8]	CVPR'25	17.73	79.95	1.84	5.75	6.15	5.69	4.01	3.19	2.59	0.015
Dispider [29]	CVPR'25	6.70	0.69	30.64	1.36	1.94	1.64	4.60	1.34	1.15	0.000
MMDuet [38]	EMNLP'25	3.81	16.99	9.79	3.13	1.87	2.99	**5.74**	2.25	1.52	0.001
LiveStar-8B [44]	NeurIPS'25	1.86	16.69	3.03	2.52	3.12	5.91	3.89	2.44	1.96	0.001
StreamingVLM [42]	ICLR'26	14.64	65.47	1.16	5.97	6.43	4.73	4.01	3.10	2.43	0.008
MMDuet2 [39]	ICLR'26	3.71	26.18	5.03	3.87	3.40	5.19	4.10	2.33	1.89	0.001
Proact-VL$_{\text{LiveCC}}$ [43]	ICML'26	16.39	83.20	0.34	5.75	5.54	5.54	4.15	3.18	2.38	0.008
Proact-VL$_{\text{Qwen3}}$ [43]	ICML'26	17.73	**83.21**	0.34	5.79	5.49	5.63	4.54	3.38	2.48	0.012
Trained on our data, human program											
LiveCC-7B-FT† [8]	CVPR'25	14.85	13.41	16.22	2.54	3.32	3.26	5.39	2.59	1.53	0.007
Proact-VL$_{\text{Qwen3}}$-FT† [43]	ICML'26	18.25	83.08	0.34	5.39	4.20	4.52	5.04	3.10	2.11	0.014
Trained on our data, self-directed program											
DirCast (ours)		**29.21**	78.22	**0.30**	**6.57**	**6.50**	**5.92**	4.97	**4.00**	**2.95**	**0.024**

TimeDiff of 0.30 seconds, and leads all three live delivery scores, the continuity and substance of the final transcript, and PAUC, with 0.024 against 0.015 for the best baseline. The two remaining metrics instead reward an extreme speaking rate. The casters speak in 73.6 percent of the seconds, so speaking in every second already reaches an F1 of 84.8. Proact-VL speaks in 99.9 percent and obtains the highest F1 of 83.21, while DirCast speaks in 86.6 percent and obtains 78.22. Fidelity favors the opposite extreme. MMDuet speaks in 10.1 percent and so makes few claims that could be wrong. Among models that speak in more than half of the seconds, DirCast has the highest fidelity at 4.97, ahead of Proact-VL at 4.54.

5.3 Ablation Studies

Table 3 Joint versus separate models. TD is TimeDiff.

		Cost		Dir.	Commentary		
Director	Comm.	GB↓	Time↓	Top-1↑	CC↑	F1↑	TD↓
Dir-only	Com-only	37.8	0.53	25.17	24.36	76.73	0.31
Ours	Com-only	37.8	0.56	**26.56**	29.04	76.64	0.31
Ours	Ours	**18.9**	**0.32**	**27.00**	**29.21**	**78.54**	**0.30**
Human	Com-only	–	–	–	25.67	76.23	0.33
Human	Ours	–	–	–	27.84	77.48	**0.30**

One model or two. Table 3 pairs DirCast with copies trained for a single role, each commentator speaking over its director's program. The joint model is better at both roles. It directs with a Top-1 of 27.00 against 25.17, wins 29.21 percent of comparisons against 24.36, and halves the memory to 18.9 GB by sharing weights. To check that the gain does not come from a better program, we compare both commentators on the human program, where the joint model still wins 27.84 percent against 25.67. Learning to direct thus makes the model a better commentator.

What makes a director. Table 4 ablates the directing components. Removing the hold-or-switch pointer costs the most, dropping Top-1 from 27.00 to 21.65 (a relative loss of about 20%) and Any-vis. from 41.80 to 35.93, which confirms that an explicit hold-or-switch decision is the backbone of the director. Without it, the director must re-select a camera from scratch at every step, which likely leads to unmotivated switches. Removing state tokens or program rollouts also hurts, dropping Top-1 to 22.59 and 22.90 and Any-vis. to 37.98 and 38.01, showing that game state complements vision and that training under self-induced states matters. Program rollouts help the commentary as well: without them CC falls from 29.21 to 28.35, since a

Table 4 Component ablations on the full test split.

	Directing		Commentary	
Variant	Top-1 ↑	Any-vis. ↑	CC ↑	TimeDiff ↓
DirCast	**27.00**	41.80	**29.21**	**0.30**
w/o hold-or-switch pointer	21.65	35.93	–	0.31
w/o state tokens	22.59	37.98	**29.21**	0.32
w/o program rollouts	22.90	38.01	28.35	**0.30**
w/o event-anchored targets	26.10	**43.06**	–	0.32

commentator that never sees its own drifted program is less prepared for the shots the director actually produces. Removing state tokens, by contrast, leaves CC unchanged at 29.21, so their benefit is confined to directing. Without event-anchored targets, Top-1 is comparable but Any-vis. rises from 41.80 to 43.06, a trade-off between matching the human director and covering every visible event on air. Since Top-1 is our primary measure of agreement with human directors, we keep event-anchored targets in the full model. Commentary metrics are otherwise stable throughout, with TimeDiff staying within 0.30–0.32, so these components mainly affect directing.

5.4 Further Analysis

Table 5 Streaming eight 288-second windows on one GPU. Time: compute per video second; KV: final cache length.

	Time (s)						
Context	Mean ↓	P95 ↓	Last 36 s ↓	KV ↓	GB ↓	Top-1 ↑	Speech
Every view kept	0.60	1.05	0.86	92.6k	32.0	23.7	67.2
Program memory	0.42	0.68	0.55	47.6k	25.1	24.6	68.1
+ 72 s horizon	0.36	0.58	0.38	12.8k	19.8	24.6	70.1
+ 36 s horizon	**0.29**	**0.52**	**0.29**	**6.9k**	**18.9**	**25.1**	76.7

Long streams. Table 5 streams eight continuous windows of 288 seconds, eight times the length of a training clip. Keeping every feed view doubles the cache to 92.6k tokens and pushes the slowest seconds beyond real time, with a P95 cost of 1.05 seconds per video second. Program memory halves the growth to 47.6k tokens. A 72-second horizon cuts the cache further to 12.8k tokens and the P95 cost to 0.58 seconds. A horizon that drops every token older than 36 seconds, the length of a training clip, keeps the cache at 6.9k tokens, memory at 18.9 GB, and the cost per second constant at 0.29 seconds on one GPU. The shorter context does not hurt direction, as Top-1 accuracy rises from 23.7 to 25.1. The horizon also keeps the commentary alive late in the stream, where the speaking rate otherwise decays, lifting speech from 67.2 to 76.7. Since every test clip is 36 seconds long, the horizon leaves all results in Tables 1 and 2 unchanged.

Scaling with training data. We train DirCast on 5, 10 and 20 matches and evaluate the director on the same test split (Figure 3). Top-1 counts a second as correct when the model airs the same feed as the human director, and Any-vis. when the chosen feed shows the event at all. Going from 5 to 10 matches gives small gains of 0.98 in Any-vis. and 0.56 in Top-1, while doubling to 20 gives 5.56 and 6.36, reaching 41.80 and 27.00. The jump suggests that the director needs to see enough fights, teams and strategies before its choices generalize to unseen matches. The curves show no sign of saturation, and new matches are cheap to collect since each needs only its camera feeds and the aired broadcast, with no manual labels, so scaling to hundreds of matches is a natural next step. Because the subsets are nested and the test split is fixed, the rising curve reflects added training variety rather than an easier evaluation.

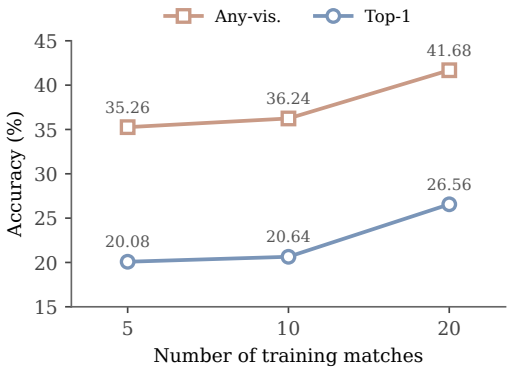


Figure 3 Accuracy vs. training matches.

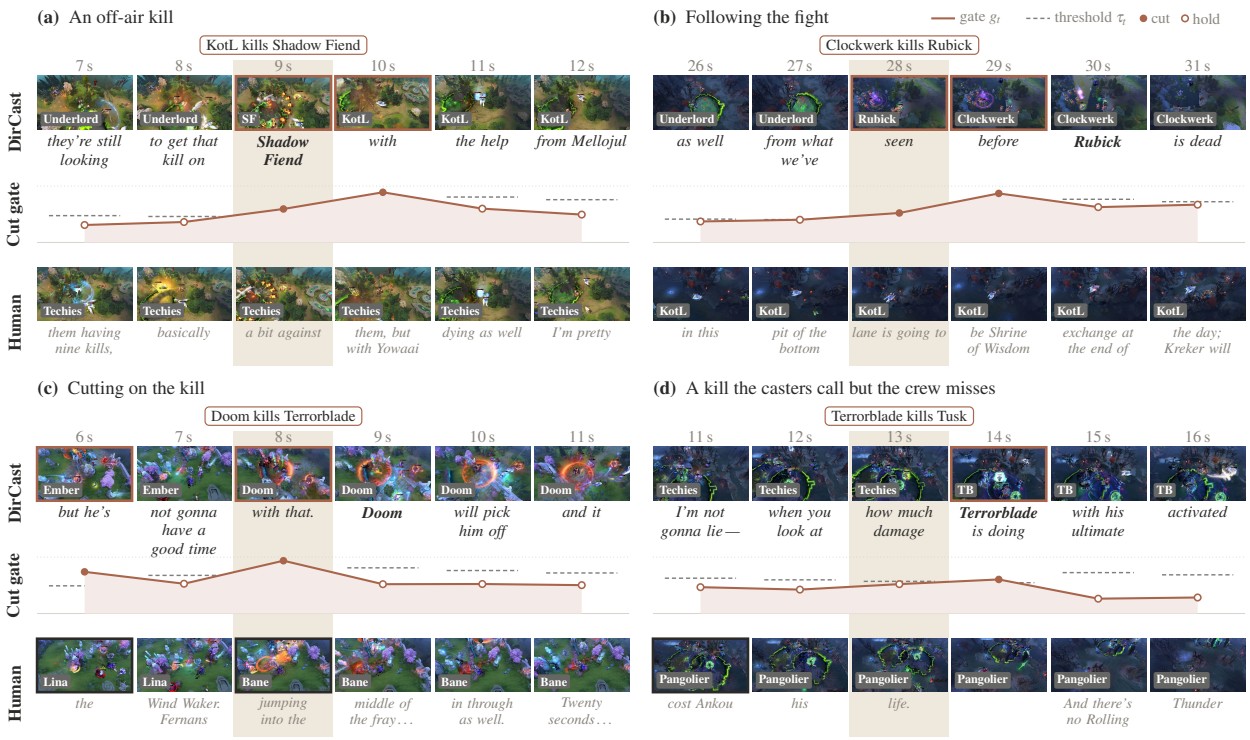


Figure 4 Four kills that DirCast shows and the human crew misses. Each panel shows DirCast's program and commentary (top), the cut gate against its threshold (middle), and the human broadcast (bottom). Framed shots are cuts, and the shaded column marks the kill: (a) KotL kills Shadow Fiend off-air, (b) Clockwerk kills Rubick, (c) Doom kills Terrorblade, and (d) Terrorblade kills Tusk.

Qualitative analysis. Figure 4 shows four of the 19 off-air kills that DirCast puts on air and names, chosen among those where the human crew shows an uninvolved hero. In each case the cut gate stays below its threshold while the action is elsewhere, so DirCast holds its shot; in the second of the kill the gate rises above the threshold and DirCast cuts to the fight, often first to the victim and then to the killer as it moves. The commentary names a hero of the fight within about two seconds of the kill: Shadow Fiend as KotL takes him down off-air in (a), Rubick as Clockwerk finishes him while DirCast follows the fight in (b), Doom as he picks off Terrorblade on the kill in (c), and Terrorblade as his ultimate kills Tusk in (d). After each cut the threshold rises, so DirCast holds the new shot instead of flickering between feeds. The human crew misses all four kills, staying on an uninvolved hero such as Techies in (a), KotL in (b), Bane in (c) and Pangolier in (d); in (d) the casters call the kill although the broadcast never shows it, so the official feed shows none of the deaths as they happen.

6 Conclusion

We introduced live esports broadcast production, where a model receives every player's camera feed and must choose the on-air shot and commentate using only past observations. DirCast addresses both tasks with a single streaming video language model. A hold-or-switch director selects the on-air shot, commentary follows its choice, and program-only context bounds the cost of long streams while enabling real-time operation on one GPU. Trained on human broadcasts with perturbed programs and event-anchored targets, DirCast outperforms rule-based and zero-shot directors including GPT-5, shows more kills on air than human crews, and wins more comparisons than eleven streaming commentators watching human broadcasts. Together, these results establish the feasibility of unified, real-time esports production with a single model.

Acknowledgments

This study was carried out using the TSUBAME4.0 supercomputer at Institute of Science Tokyo.

References

[1] Ido Arev, Hyun Soo Park, Yaser Sheikh, Jessica Hodgins, and Ariel Shamir. Automatic editing of footage from multiple social cameras. *ACM Transactions on Graphics (TOG)*, 33(4):1–11, 2014.

[2] Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

[3] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. *Advances in neural information processing systems*, 28, 2015.

[4] Jianhui Chen, Hoang M Le, Peter Carr, Yisong Yue, and James J Little. Learning online smooth predictors for realtime camera planning using recurrent decision trees. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4688–4696, 2016.

[5] Jianhui Chen, Lili Meng, and James J Little. Camera selection for broadcasting soccer games. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 427–435. IEEE, 2018.

[6] Jianhui Chen, Keyu Lu, Sijia Tian, and Jim Little. Learning sports camera selection from internet videos. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1682–1691. IEEE, 2019.

[7] Joya Chen, Zhaoyang Lv, Shiwei Wu, Kevin Qinghong Lin, Chenan Song, Difei Gao, Jia-Wei Liu, Ziteng Gao, Dongxing Mao, and Mike Zheng Shou. Videollm-online: Online video large language model for streaming video. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18407–18418. IEEE, 2024.

[8] Joya Chen, Ziyun Zeng, Yiqi Lin, Wei Li, Zejun Ma, and Mike Zheng Shou. Live: Learning video llm with streaming speech transcription at scale. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 29083–29095. IEEE, 2025.

[9] Shangzhe Di, Zhelun Yu, Guanghao Zhang, Haoyuan Li, Hao Cheng, Bolin Li, Wanggui He, Fangxun Shu, and Hao Jiang. Streaming video question-answering with in-context video kv-cache retrieval. In *International Conference on Learning Representations*, volume 2025, pages 42115–42127, 2025.

[10] Silvio Giancola, Mohieddine Amine, Tarek Dghaily, and Bernard Ghanem. Soccernet: A scalable dataset for action spotting in soccer videos. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1792–179210. IEEE, 2018.

[11] Kristen Grauman, Andrew Westbury, Lorenzo Torresani, Kris Kitani, Jitendra Malik, Triantafyllos Afouras, Kumar Ashutosh, Vijay Baiyya, Siddhant Bansal, Bikram Boote, et al. Ego-exo4d: Understanding skilled human activity from first-and third-person perspectives. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 19383–19400, 2024.

[12] Bo He, Hengduo Li, Young Kyun Jang, Menglin Jia, Xuefei Cao, Ashish Shah, Abhinav Shrivastava, and Ser-Nam Lim. Ma-lmm: Memory-augmented large multimodal model for long-term video understanding. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13504–13514. IEEE, 2024.

[13] Hou-Ning Hu, Yen-Chen Lin, Ming-Yu Liu, Hsien-Tzu Cheng, Yung-Ju Chang, and Min Sun. Deep 360 pilot: Learning a deep agent for piloting through 360 sports videos. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1396–1405. IEEE, 2017.

[14] Tatsuya Ishigaki, Goran Topić, Yumi Hamazono, Hiroshi Noji, Ichiro Kobayashi, Yusuke Miyao, and Hiroya Takamura. Generating racing game commentary from vision, language, and structured data. In *Proceedings of the 14th International Conference on Natural Language Generation*, pages 103–113, 2021.

[15] Harsh Jhamtani, Varun Gangal, Eduard Hovy, Graham Neubig, and Taylor Berg-Kirkpatrick. Learning to generate move-by-move commentary for chess games from large-scale social forum data. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1661–1671, 2018.

[16] Alex M Lamb, Anirudh Goyal Alias Parth Goyal, Ying Zhang, Saizheng Zhang, Aaron C Courville, and Yoshua Bengio. Professor forcing: A new algorithm for training recurrent networks. *Advances in neural information processing systems*, 29, 2016.

[17] Mackenzie Leake, Abe Davis, Anh Truong, and Maneesh Agrawala. Computational video editing for dialogue-driven scenes. *ACM Trans. Graph.*, 36(4):130–1, 2017.

[18] Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Peiyuan Zhang, Yanwei Li, Ziwei Liu, et al. Llava-onevision: Easy visual task transfer. *arXiv preprint arXiv:2408.03326*, 2024.

[19] Yanwei Li, Chengyao Wang, and Jiaya Jia. Llama-vid: An image is worth 2 tokens in large language models. In *European conference on computer vision*, pages 323–340. Springer, 2024.

[20] Bin Lin, Yang Ye, Bin Zhu, Jiaxi Cui, Munan Ning, Peng Jin, and Li Yuan. Video-llava: Learning united visual representation by alignment before projection. In *Proceedings of the 2024 conference on empirical methods in natural language processing*, pages 5971–5984, 2024.

[21] Junming Lin, Zheng Fang, Chi Chen, Haoxuan Cheng, Zihao Wan, Fuwen Luo, Ziyue Wang, Peng Li, Yang Liu, and Maosong Sun. Streamingbench: Assessing the gap for mllms to achieve streaming video understanding. In *ICASSP 2026-2026 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 12147–12151. IEEE, 2026.

[22] Shuming Ma, Lei Cui, Damai Dai, Furu Wei, and Xu Sun. Livebot: Generating live video comments based on visual and textual contexts. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6810–6817, 2019.

[23] Sagnik Majumder, Tushar Nagarajan, Ziad Al-Halah, Reina Pradhan, and Kristen Grauman. Which viewpoint shows it best? language for weakly supervising view selection in multi-view instructional videos. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 29016–29028. IEEE, 2025.

[24] Hassan Mkhallati, Anthony Cioppa, Silvio Giancola, Bernard Ghanem, and Marc Van Droogenbroeck. Soccernet-caption: Dense video captioning for soccer broadcasts commentaries. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 5074–5085. IEEE, 2023.

[25] Volodymyr Mnih, Nicolas Heess, Alex Graves, and Koray Kavukcuoglu. Recurrent models of visual attention. *Advances in neural information processing systems*, 27, 2014.

[26] Junbo Niu, Yifei Li, Ziyang Miao, Chunjiang Ge, Yuanhang Zhou, Qihao He, Xiaoyi Dong, Haodong Duan, Shuangrui Ding, Rui Qian, et al. Ovo-bench: How far is your video-llms from real-world online video understanding? In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18902–18913. IEEE, 2025.

[27] Ji Qi, Jifan Yu, Teng Tu, Kunyu Gao, Yifan Xu, Xinyu Guan, Xiaozhi Wang, Bin Xu, Lei Hou, Juanzi Li, et al. Goal: A challenging knowledge-grounded video captioning benchmark for real-time soccer commentary generation. In *Proceedings of the 32nd ACM international conference on information and knowledge management*, pages 5391–5395, 2023.

[28] Rui Qian, Xiaoyi Dong, Pan Zhang, Yuhang Zang, Shuangrui Ding, Dahua Lin, and Jiaqi Wang. Streaming long video understanding with large language models. *Advances in Neural Information Processing Systems*, 37:119336–119360, 2024.

[29] Rui Qian, Shuangrui Ding, Xiaoyi Dong, Pan Zhang, Yuhang Zang, Yuhang Cao, Dahua Lin, and Jiaqi Wang. Dispider: Enabling video llms with active real-time interaction via disentangled perception, decision, and reaction. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 24045–24055. IEEE, 2025.

[30] Jiayuan Rao, Haoning Wu, Chang Liu, Yanfeng Wang, and Weidi Xie. Matchtime: Towards automatic soccer game commentary generation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 1671–1685, 2024.

[31] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

[32] Enxin Song, Wenhao Chai, Guanhong Wang, Yucheng Zhang, Haoyang Zhou, Feiyang Wu, Haozhe Chi, Xun Guo, Tian Ye, Yanting Zhang, et al. Moviechat: From dense token to sparse memory for long video understanding. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18221–18232. IEEE, 2024.

[33] Yu-Chuan Su, Dinesh Jayaraman, and Kristen Grauman. Pano2vid: Automatic cinematography for watching 360 videos. In *Asian Conference on Computer Vision*, pages 154–171. Springer, 2016.

[34] Tsunehiko Tanaka and Edgar Simo-Serra. Lol-v2t: Large-scale esports video description dataset. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 4552–4561. IEEE, 2021.

[35] Haibo Wang, Bo Feng, Zhengfeng Lai, Mingze Xu, Shiyu Li, Weifeng Ge, Afshin Dehghan, Meng Cao, and Ping Huang. Streambridge: Turning your offline video large language model into a proactive streaming assistant. *Advances in Neural Information Processing Systems*, 38:132332–132359, 2026.

[36] Miao Wang, Guo-Wei Yang, Shi-Min Hu, Shing-Tung Yau, Ariel Shamir, et al. Write-a-video: computational video montage from themed text. *ACM Trans. Graph.*, 38(6):177–1, 2019.

[37] Weiyun Wang, Zhangwei Gao, Lixin Gu, Hengjun Pu, Long Cui, Xingguang Wei, Zhaoyang Liu, Linglin Jing, Shenglong Ye, Jie Shao, et al. Internvl3. 5: Advancing open-source multimodal models in versatility, reasoning, and efficiency. *arXiv preprint arXiv:2508.18265*, 2025.

[38] Yueqian Wang, Xiaojun Meng, Yuxuan Wang, Jianxin Liang, Jiansheng Wei, Huishuai Zhang, and Dongyan Zhao. Videollm knows when to speak: Enhancing time-sensitive video comprehension with video-text duet interaction format. In *EMNLP (Findings)*, pages 6338–6359, 2025.

[39] Yueqian Wang, Songxiang Liu, Disong Wang, Nuo Xu, Guanglu Wan, Huishuai Zhang, and Dongyan Zhao. Mmduet2: Enhancing proactive interaction of video mllms with multi-turn reinforcement learning. In *International Conference on Learning Representations*, volume 2026, pages 23257–23270, 2026.

[40] Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations*, volume 2024, pages 21875–21895, 2024.

[41] Mingze Xu, Mingfei Gao, Zhe Gan, Hong-You Chen, Zhengfeng Lai, Haiming Gang, Kai Kang, and Afshin Dehghan. Slowfast-llava: A strong training-free baseline for video large language models. *arXiv preprint arXiv:2407.15841*, 2024.

[42] Ruyi Xu, Guangxuan Xiao, Yukang Chen, Liuning He, Kelly Peng, Yao Lu, and Song Han. Streamingvlm: Real-time understanding for infinite video streams. In *International Conference on Learning Representations*, volume 2026, pages 61463–61475, 2026.

[43] Weicai Yan, Yuhong Dai, Qi Ran, Haodong Li, Wang Lin, Tao Jin, Xing Xie, Hao Liao, and Jianxun Lian. Proact-vl: A proactive videollm for real-time ai companions. *arXiv preprint arXiv:2603.03447*, 2026.

[44] Zhenyu Yang, Kairui Zhang, Yuhang Hu, Bing Wang, Shengsheng Qian, Bin Wen, Fan Yang, Tingting Gao, Weiming Dong, and Changsheng Xu. Livestar: Live streaming assistant for real-world online video understanding. *Advances in Neural Information Processing Systems*, 38:31266–31304, 2026.

[45] Dawei Zhang, Sixing Wu, Yao Guo, and Xiangqun Chen. Moba-e2c: Generating moba game commentaries via capturing highlight events from the meta-data. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 4545–4556, 2022.

[46] Haoji Zhang, Yiqin Wang, Yansong Tang, Yong Liu, Jiashi Feng, Jifeng Dai, and Xiaojie Jin. Flash-vstream: Memory-based real-time understanding for long video streams. *arXiv preprint arXiv:2406.08085*, 2024.

[47] Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, et al. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in neural information processing systems*, 36:34661–34710, 2023.

[48] Zhihao Zhang, Feiqi Cao, Yingbin Mo, Yiran Zhang, Josiah Poon, and Caren Han. Game-mug: Multimodal oriented game situation understanding and commentary generation dataset. *arXiv preprint arXiv:2404.19175*, 2024.

Appendix

A Dataset Construction

This section describes how the dataset is built: the director data (Section A.1), the commentary (Section A.2), and the cleaning applied to both (Section A.3).

A.1 Director Data

Source data. We build a multi-camera director dataset from 27 professional *Dota 2* matches (EPL Masters and BLAST Slam qualifiers, 2026). For each match we use two public sources: the official broadcast video (VOD), which is the only record of the human director's output and of the casters' voices, and the official replay file (`.dem`), which contains the complete game state at 30 ticks per second. Player and hero metadata are taken from OpenDota.

Broadcast-to-replay alignment. The broadcast and the replay share the in-game clock shown on the HUD. We OCR the HUD clock from the VOD and fit a linear mapping between VOD time and the game clock, with a residual of about 0.3 s. Because broadcasts often cover only part of a match, and some archive records point to the wrong game, we additionally OCR the hero names on the HUD and keep only VOD stretches whose hero set matches the replay. Game pauses require special care: replay ticks and the broadcast keep running during a pause while the game clock stops, so the mapping is defined on replay ticks with the paused ticks subtracted. We validate the alignment with a consistency check: heroes whose name tags are visible together in a broadcast frame must be close to each other in the replay at the mapped tick. All 27 matches pass this check at zero time offset.

Candidate camera feeds. Candidate feeds are rendered with the Dota 2 client itself. Each replay is played back in a headless X server and recorded by ten spectator clients in parallel, one per player, each in the client's *Hero Chase* mode following that player's hero with the HUD hidden. Clients are driven through the console interface; capture starts and ends at fixed replay ticks and every frame is tagged with its tick, so all ten feeds are synchronized with the replay and with the broadcast. Feeds are captured at 1280×720 and 30 fps and stored at 640×360; training clips use one frame per second resized to 288×288. Each match is rendered in 300 s windows covering the broadcast-aligned part of the game.

Director labels. The label for each second is the hero the broadcast is looking at, which maps to one of the ten feeds. We obtain it from the broadcast frame itself: the spectator HUD prints a name tag above every visible hero, which we OCR and match to the match roster, and the HUD also shows the currently selected unit. The label is the selected hero if it is on screen, otherwise the visible hero nearest the frame centre, with a 2 s hysteresis on switches and up to 4 s of carry-over when no tag is readable. Every label records which of these cases produced it, which the cleaning in Section A.3 uses. Every second also receives a broadcast state (live, replay, production, paused, no coverage) derived from the clock OCR and a gameplay cue. We additionally store the set of all heroes on screen in each second, which supports an *any-of-visible* evaluation, since several feeds can show the same scene.

A.2 Commentary

Audio and transcription. For every rendered window we cut the broadcast audio at the same aligned VOD times as the director labels, so the commentary shares the time axis of the feeds and the labels, and resample it to 16 kHz mono. The casters' speech is transcribed with Whisper (large-v3-turbo) in English with voice activity detection, beam size 5 and word-level timestamps. Each window is decoded without conditioning on the previous text, which avoids repetition loops over long audio, and with a vocabulary prompt that lists the heroes and players of the match, which biases the decoder towards the names the casters use. Every word is assigned to the second in which it starts, which gives the per-second commentary targets. Of the 2,096 training clips, 1,984 have broadcast audio, and about 8% of the transcribed words have an ASR confidence below 0.5.

Polishing. Raw ASR misspells hero and player names and keeps fillers and repetitions. Such text would teach the model wrong names and would penalise correct names at evaluation. We therefore follow the data pipeline of Proact-VL [43], which merges ASR segments into sentences and polishes them with a language model. Consecutive Whisper segments are merged into a sentence until end punctuation, a pause of more than 1 s, 30 words or 12 s. Sentences are then rewritten in batches of twelve by GPT-5-mini with a match-specific prompt. The prompt lists both teams and their players and heroes, and asks the model to correct hero, player, item and ability names, to remove fillers and repetitions, and to keep the casters' own words, word order and meaning. Sentences that the casters leave unfinished stay unfinished, and the model may not add who did what, numbers or outcomes. Proact-VL uses DeepSeek-V3.2 for this step. We use GPT-5-mini because GPT-4o is the opponent and GPT-5 the judge of the commentary evaluation, so neither of them sees the references beforehand.

Guard and timing. A rewrite is accepted only if its word count lies between 70% and 140% of the ASR sentence and at least half of its words appear in the ASR sentence; otherwise we keep the ASR text with fillers removed. This guard prevents the language model from inventing content. Polishing keeps the timing of speech: the polished words of a sentence take the start times of the original words in order and are bucketed into seconds by the same rule as the raw transcript. Polishing changes the text of 38% of the seconds with speech and removes 2% of the words, mostly fillers and repetitions.

Use. The polished commentary is the training target of DirCast and of the two fine-tuned baselines, and it is the reference for every model at test time, as in the Proact-VL protocol.

A.3 Cleaning

Table 7 follows every clip-second through the cleaning steps.

Broadcast state. Only seconds in which the broadcast shows live gameplay are director decisions. Replays, production shots such as the desk or the drafting screen, pauses and seconds without broadcast coverage are removed, which keeps 87% of the clip-seconds.

Label quality. Labels that come from the HUD selection alone, without the selected hero on screen, are dropped, since only 52% of them agree with the hero positions in the replay. The remaining labels form the training mask.

Verification. Carried-over labels are the noisiest, because the broadcast may cut while no name tag is readable. We verify all 15,951 carried-over seconds with a vision–language model (GPT-6 Astra). It sees the broadcast frame and four candidate feeds, the labelled feed among them, and answers which of them shows the same scene; a label is confirmed if the answer is the labelled feed. 62% are confirmed, and the rejected seconds are removed from the training mask, which gives the *verified* mask used for training. On a manually audited sample the model's verdicts agree with the human audit in 43 of 45 cases.

Strict evaluation subset. Evaluation uses only labels that are observed on screen in that very second, that is, the selected hero with its tag visible or the visible hero nearest the frame centre, without hysteresis or carry-over. This *strict* mask keeps 69% of the live test seconds.

Commentary. Commentary is cleaned by the polishing and guard described above. Windows without broadcast audio keep their director labels but contribute no commentary targets.

Splits. Matches are split as a whole into 20 training, 3 validation and 4 test matches, so no game, broadcast or caster utterance is shared between training and test. The validation matches are used to calibrate the base thresholds of the decoding rules.

A.4 Statistics

The dataset contains 15.2 hours of aligned ten-feed footage, of which 13.1 hours are live broadcast and 10.3 hours carry verified labels (7.9 hours in the strict subset). It is packaged as 2,912 clips of 36 s (stride 18 s), split as in Table 6. Casters speak in 64% of the training clip-seconds and 72% of the test clip-seconds. The director switches feeds every few seconds, so a persistence baseline that repeats the previous label reaches about 80% per-second accuracy; we therefore report switch-second accuracy separately.

Split	Matches	Clips	Verified clip-s	Strict clip-s	Speech	Words
Train	20	2,096	50,283	35,815	64%	142.0k
Val	3	320	9,627	7,357	50%	16.3k
Test	4	496	14,630	11,283	72%	38.0k

Table 6 Dataset statistics. Clips are 36 s long with a stride of 18 s, so clip-seconds and words count overlapping clips; unique footage totals 15.2 h. Speech is the share of clip-seconds with polished commentary.

Clip-seconds	Train	Val	Test
All	75,456	11,520	17,856
Live broadcast	63,374	11,143	16,238
without HUD-only labels (training mask)	59,574	10,977	15,894
after verification (verified mask)	**50,283**	9,627	14,630
Observed on screen (strict mask)	35,815	7,357	**11,283**

Table 7 Cleaning funnel. Bold marks the seconds used for training and for evaluation. Clip-seconds count overlapping clips.

B Additional qualitative examples

We show twenty more off-air kills that DirCast puts on air and names, ten in Figure 5 and ten in Figure 6, drawn from the test matches and across the random seeds used in the main experiments. The examples cover a range of heroes and situations, from single-target pickoffs to crowded team fights, and each is chosen among the off-air kills where the human crew shows an uninvolved hero, so they test the director on moments it has to find on its own.

Each panel follows the format of Figure 4. The label at the top names the kill and the seed. The top row is DirCast's program, one on-air thumbnail per second with the words it speaks printed underneath; the middle row plots the cut gate against its threshold, with markers for the seconds it cuts and the seconds it holds; and the bottom row is the human broadcast over the same seconds. The shaded column marks the second of the kill.

Across the examples DirCast shows the behaviour described in Section 5.4: it holds its shot while the gate stays below the threshold, cuts to the fight as the gate rises in the second around the kill, often moving from the victim to the killer as the fight develops, and names a hero of the fight within about two seconds. After the cut the threshold rises, so it holds the new shot instead of flickering between feeds. By construction the human crew is on an uninvolved hero in every panel, and it shows the kill late or not at all, so these decisive seconds never reach the official broadcast even though DirCast puts them on air and calls them.

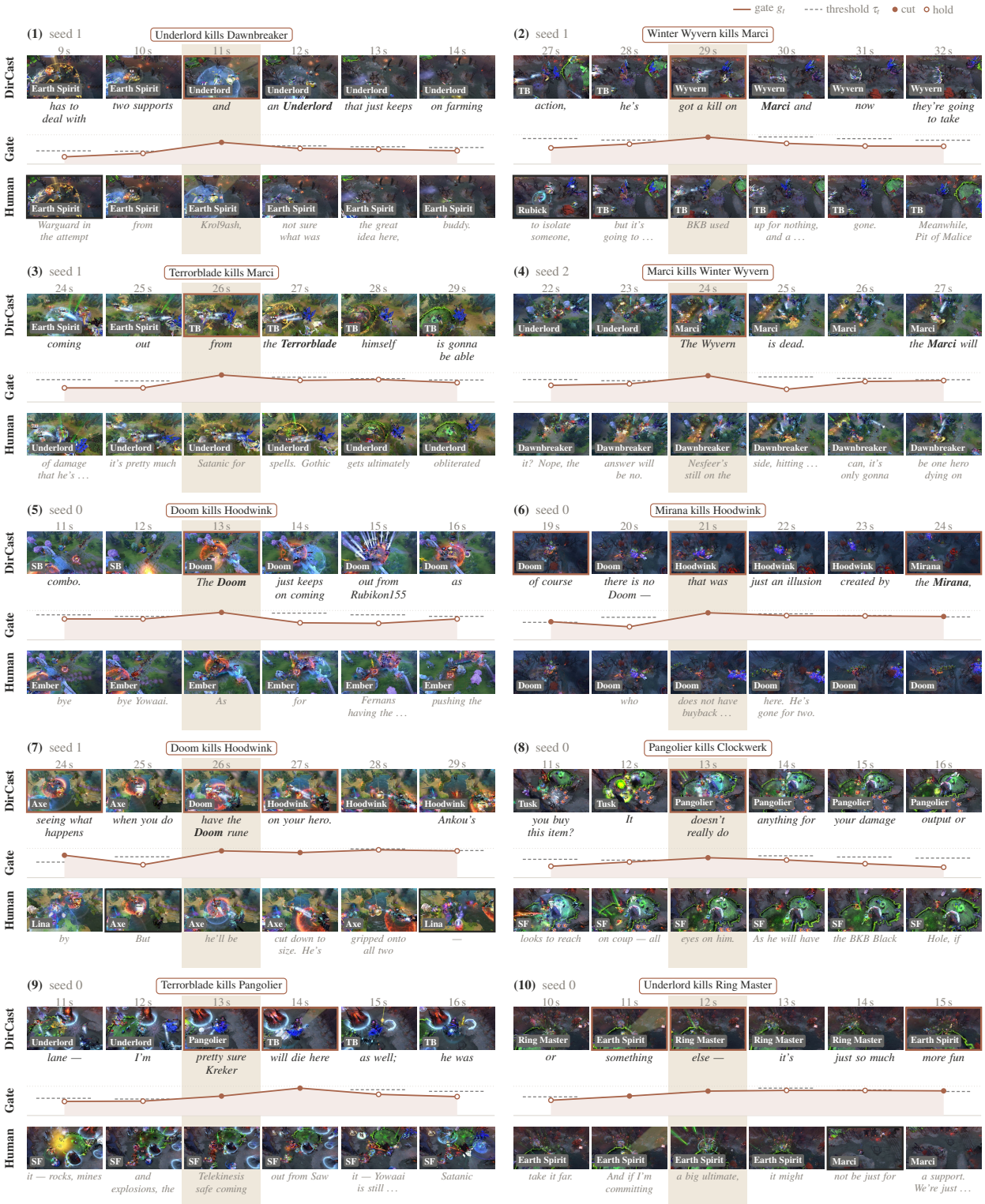


Figure 5 Additional qualitative examples of off-air kills that DirCast shows and names (1–10 of 20).

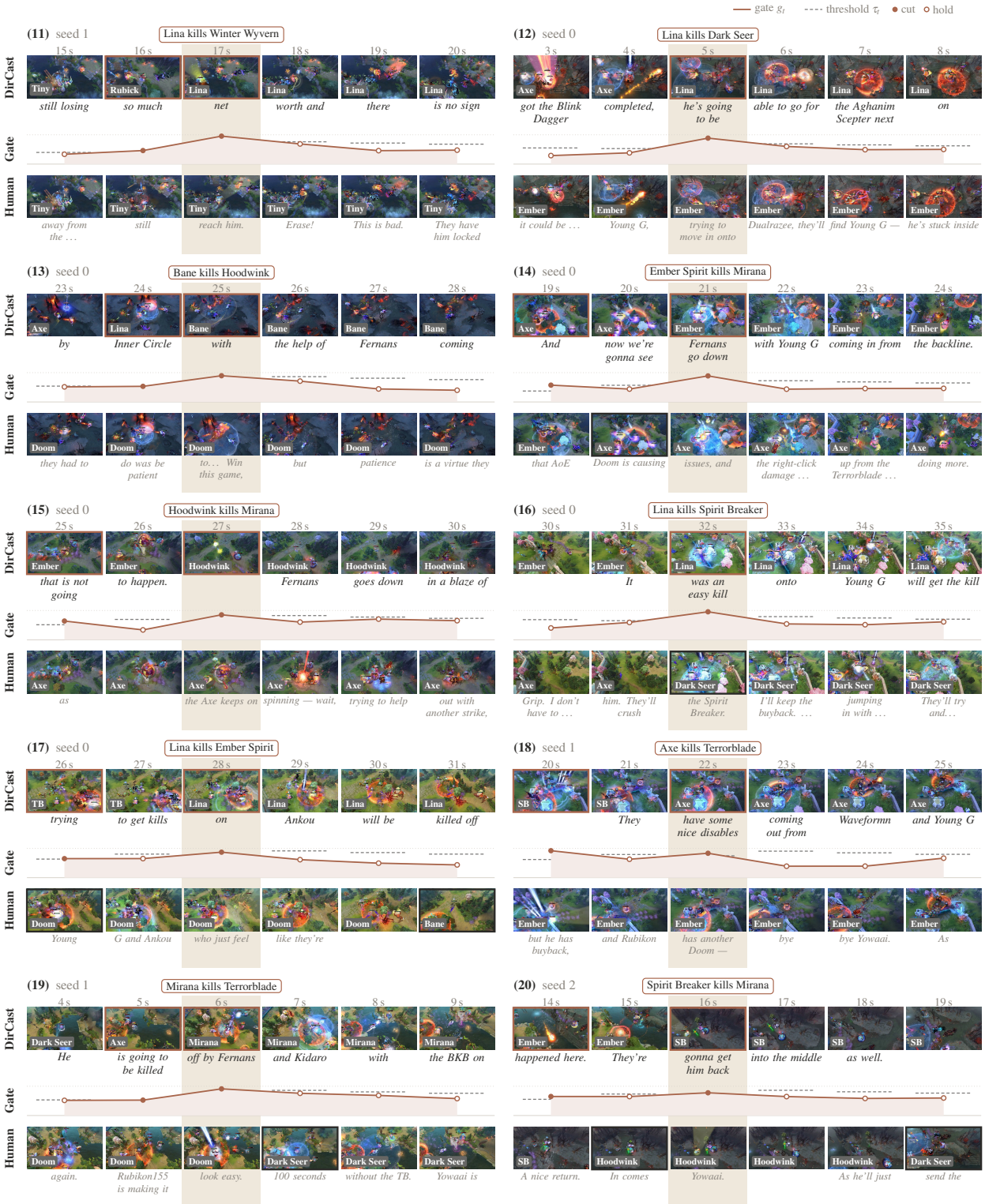


Figure 6 Additional qualitative examples of off-air kills that DirCast shows and names (11–20 of 20).