As humans, we navigate the world through all our senses, using perceptual input from each one to correct the others. We introduce MERLOT Reserve, a model that represents videos jointly over time – through a new training objective that learns from audio, subtitles, and video frames. Given a video, we replace snippets of text and audio with a MASK token; the model learns by choosing the correct masked-out snippet. Our objective learns faster than alternatives, and performs well at scale: we pretrain on 20 million YouTube videos.

Empirical results show that MERLOT Reserve learns strong representations about videos through all constituent modalities. When finetuned, it sets a new state-of-the-art on both VCR and TVQA, outperforming prior work by 5% and 7% respectively. Ablations show that both tasks benefit from audio pretraining – even VCR, a QA task centered around images (without sound). Moreover, our objective enables out-of-the-box prediction, revealing strong multimodal commonsense understanding. In a fully zero-shot setting, our model obtains competitive results on four video understanding tasks, even outperforming supervised approaches on the recently proposed Situated Reasoning (STAR) benchmark.

We analyze why incorporating audio leads to better vision-language representations, suggesting significant opportunities for future research. We conclude by discussing ethical and societal implications of multimodal pretraining.

1. Introduction

The world around us is dynamic. We experience and learn from it using all of our senses, reasoning over them temporally through multimodal script knowledge [98, 126]. Consider Figure 1, which depicts someone cooking popcorn. From the images and dialogue alone, we might be able to imagine what sounds of the scene are: the process might begin with raw kernels scattering in an empty, metallic pot, and end with the dynamic ‘pops’ of popcorn expanding, along with the jiggling of a metal around the stove.

Predicting this sound is an instance of learning from reentry: where time-locked correlations enable one modality to educate others. Reentry has been hypothesized by developmental psychologists to be crucial for how we as humans learn visual and world knowledge, much of it without need for an explicit teacher [88, 35, 19, 100]. Yet, we ask – can we build machines that likewise learn vision, language, and sound together? And can this paradigm enable learning neural script knowledge, that transfers to language-and-vision tasks, even those without sound?

In this work, we study these questions, and find that the answers are ‘yes.’ We introduce a new model that learns self-supervised representations of videos, through all their modalities (audio, subtitles, vision). We dub our model MERLOT Reserve, henceforth Reserve for short.

\[ \text{Add a third of a cup of popcorn} \]
\[ \text{Now turn the heat on high} \]
\[ \text{Add a lid, and then} \]
\[ *\text{sizzling}*, *\text{pouring sound}*, *\text{lid clinking}*, \]
\[ \text{**jiggling, popcorn popping**} \]

Figure 1: MERLOT Reserve learns multimodal neural script knowledge representations of video – jointly reasoning over video frames, text, and audio. Our model is pretrained to predict which snippet of text (and audio) might be hidden by the MASK. This task enables it to perform well on a variety of vision-and-language tasks, in both zero-shot and finetuned settings.
Our model differs from past work that learns from audio-image pairs [53, 70], from subtitled videos [105, 126], or from static images with literal descriptions [106, 20, 91]. Instead, we learn joint representations from all modalities of a video, using each modality to teach others. We do this at scale, training on over 20 million YouTube videos.

We introduce a new contrastive masked span learning objective to learn script knowledge across modalities. It generalizes and outperforms a variety of previously proposed approaches (e.g. [29, 106, 91, 126]), while enabling audio to be used as signal. The idea is outlined in Figure 1: the model must figure out which span of text (or audio) was masked out of a video sequence. We combine our objective with a second contrastive learning approach, tailored to learning visual recognition from scratch: the model must also match each video frame to a contextualized representation of the video’s transcript [126]. Through ablations, we show that our framework enables rapid pretraining of a model and readily scales to ‘large’ transformer sizes (of 644M parameters).

Experimental results show that RESERVE learns powerful representations, useful even for tasks posed over only a subset of the studied modalities. For example, when finetuned on Visual Commonsense Reasoning [124] (a vision+language task with no audio), it sets a new state-of-the-art, outperforming models trained on supervised image-caption pairs by over 5%. It does even better on video tasks: fine-tuning without audio, it outperforms prior work on TVQA [72] by a margin of over 7% (and given TVQA audio, performance increases even further). These performance improvements do not come at the expense of efficiency: our largest model uses one-fifths the FLOPs of a VisualBERT.

RESERVE also performs well in zero-shot settings. We evaluate on four diverse benchmarks: Situated Reasoning (STAR) [119], EPIC-Kitchens [26], LSMDC-FiB [95], and MSR-VTT QA [120]. These benchmarks require visual reasoning with respective emphasis on temporality, future prediction, and both social and physical understanding. With no fine-tuning or supervision, our model obtains competitive performance on each. Of note, it nearly doubles [122]’s SoTA zero-shot accuracy on MSR-VTT QA, and it outperforms supervised approaches (like ClipBERT [73]) on STAR.

Finally, we investigate why, and on which training instances audio-powered multimodal pretraining particularly helps. For instance, predicting audio rewards models for recognizing dynamic state changes (like cooked popcorn) and human communication dynamics (what are people’s emotions and towards whom). Our model progressively learns these phenomena as pretraining progresses. These signals are often orthogonal to what snippets of text provide, which motivates learning from both modalities.

In summary, our key contributions are the following:

a. RESERVE, a model for multimodal script knowledge, fusing vision, audio, and text.
b. A new contrastive span matching objective, enabling our model to learn from text and audio self-supervision.
c. Experiments, ablations, and analysis, that demonstrate strong multimodal video representations.

Overall, the results suggest that learning representations from all modalities – in a time-locked, reentrant manner – is a promising direction, and one that has significant space for future work. We release code and model checkpoints at rowanzellers.com/merlotreserve.

2. Related Work

Our work brings together two active lines of research.

**Joint representations of multiple modalities.** Many language-and-vision tasks benefit from early fusion of different modalities [7]. A family of ‘VisualBERT’ models have been proposed for this: typically, these use a supervised object detector image encoder backbone, and pretrain on images paired with literal captions [106, 76, 80, 20, 123, 73]. Cross-modal interactions are learned in part through a masked language modeling (mask LM) objective [29], where sub-words are replaced with ‘MASK’, and models independently predict each subword conditioned on both images and un-masked tokens. Extensions to this paradigm include audio in the captioning setting [61] – for instance, OPT [78] was trained on image-caption pairs narrated by a text-to-speech system; yet like the other approaches, this still only learns (from audio) at the level of literal captions.

Perhaps closest to our work is MERLOT [126], which learns a joint vision-text model from web videos with automatic speech recognition (ASR). Through a combination of objectives (including a variant of mask LM), MERLOT established strong results on a variety of video QA benchmarks when finetuned. However, it lacks audio: it is limited to learning from (and representing) video frames paired with subtitles. Our proposed RESERVE, which represents and learns from audio, outperforms MERLOT.

**Co-supervision between modalities.** A common pitfall when training a joint multimodal model is that complex inter-modal interactions can be ignored during learning, in favor of simpler intra-modal interactions [50, 23, 58]. For example, when using the aforementioned mask LM objective, models often ignore visual input completely in favor of text-text interactions [13]; this issue is magnified when training on videos with ASR, as spoken language often contains redundant words [126].

As a result, a line of recent work has learned independent modality-specific encoders coupled with objectives that can’t

---

3Recent papers propose exceptions, like generating masked-out spans [21] or text [77, 116], but it is unclear whether they can outperform the VisualBert’s on vision-language tasks like VCR [124].
be shortcutted with simple intra-modal patterns. Models like CLIP learn image classification by matching images with their captions, contrastively [130, 91, 62]. Recent work has explored this paradigm for matching video frames with their transcripts [121], with their audio signal [96, 114], or both [3, 2]; these works likewise perform well on single-modality tasks like audio classification and activity recognition. These independent encoders can be combined through late fusion [96], yet late fusion is strictly less expressive than our proposed joint encoding (early fusion) approach.

Our work combines both lines of research. We train a joint model using all modalities in a video — vision, text, and audio — and use a new objective to enable co-supervision between modalities. For instance, we can learn a strong model of vision-and-text in part by using audio as supervision: when the audio is of “MASKed-out” regions of the video, models cannot easily ‘cheat’ by learning intra-modal vision-vision or text-text interactions (Figure 1). We turn to this next.

3. Model: 🎖️RESERVE

In this section, we present 🎖️RESERVE, including: our model architecture (3.1), new pretraining objectives (3.2), and pretraining video dataset (3.3). At a high level, 🎖️RESERVE represents a video by fusing its constituent modalities (vision, audio, and text from transcribed speech) together, and over time. These representations enable both finetuned and zero-shot downstream applications.

More formally, we split a video $V$ into a sequence of non-overlapping segments in time $\{s_t\}$. Each segment has:

- **a.** A frame $v_t$, from the middle of the segment,
- **b.** The ASR tokens $w_t$ spoken during the segment,
- **c.** The audio $a_t$ of the segment.

Segments default to 5 seconds in length; we discuss details of how we split videos into segments in Appendix C.

As the text $w_t$ was automatically transcribed by a model given audio $a_t$, it is reasonable to assume that it contains strictly less information content.\(^3\) Thus, for each segment $s_t$, we provide models with exactly one text or audio. We will further mask out portions of the text and audio during pretraining, to challenge models to recover what is missing.

3.1. Model architecture

An overview of 🎖️RESERVE is shown in Figure 2. Each modality is first encoded independently by a separate encoder (a Transformer [110] for images/audio; a BPE lookup table

\(^3\) Despite being derived from the audio, pretraining with text is still paramount: 1) in §3.2 we discuss how jointly modeling audio+text prevents models from shortcutting pretraining objectives via surface correlations; 2) in §4.2 we show that incorporating both transcripts and audio during fine-tuning improves performance; and 3) a textual interface to the model is required for downstream vision+language with textual inputs.

```latex
\begin{enumerate}
\item Predict MASKed text and audio
\item Joint Encoder (Transformer) for all modalities and timesteps
\item Inputs for segment $t$
\end{enumerate}
```

Figure 2: 🎖️RESERVE architecture. We provide sequence-level representations of video frames, and either words or audio, to a joint encoder. The joint encoder contextualizes over modalities and timesteps, to predict what is behind MASK for audio $\hat{a}_t$ and text $\hat{w}_t$. We supervise these predictions with independently encoded targets: $a_t$ from the audio encoder, and $w_t$ from a separate text encoder (not shown).

for text). Then, the representations will be combined with a joint encoder. All encoders are bidirectional.

**Image encoder.** We first encode each frame independently with a Vision Transformer (ViT; [34]), which represents an image of dimensions $H \times W$ as a sequence of 16×16-pixel patches. We conserve memory in the joint encoder by then performing a 2x2 pooling of the feature map, giving us a final $H/32 \times W/32$ sequence of dimension $d_h$.\(^4\)

**Audio encoder.** We split the audio in each segment $a_t$ into three equal-sized *subsegments*, for compatibility with the lengths at which we mask text (Appendix C). Each subsegment is encoded independently using an Audio Spectrogram Transformer (AST; [46]), which treats each subsegment as an $M \times T$ image to be ViT-encoded (where $M=64$ is the Mel filterbank size, and $T=30$ is time). We treat it as a one-dimensional sequence by using a patch size of $M \times 2$. We attention-pool the resulting feature map with a window size of 5, giving a final feature map of $T/10=6$.

**Text encoder.** We provide text to our model by BPE-encoding it, embedding the result with a learned table [99].

**Joint encoder.** Finally, we jointly encode all modalities (over all input video segments) using a bidirectional Transformer. We use a linear projection of the final layer’s hidden states for all objectives (e.g. $w_t$ and $a_t$).

**Contrastive targets.** Our contrastive pretraining objectives (3.2) require independently-encoded targets for each modality.\(^5\) Doing this is straightforward for the image and

\(^4\) We pool through self-attention: the query is the average of the vectors in the 2x2 grid; the keys and values are learned projections of the vectors.

\(^5\) Otherwise, a model can trivially cheat by coordinating predictions [6].
Figure 3: Contrastive span training. Given a video with all modalities temporally aligned, we \texttt{MASK} out a region of text and audio. The model must maximize its similarity \textit{only to} an independent encoding of the text \(w_t\) and audio \(a_t\).

audio encoders: we add a \texttt{CLS} to their respective inputs, and extract the final hidden state \(v_t\) or \(a_t\) at that position.\(^6\)

For text, we learn a separate bidirectional Transformer \textit{span encoder}, which computes targets \(w_t\) from a \texttt{CLS} and (shared) embedded tokens of a candidate text span. This will also enable zero-shot prediction (4.3).

\textbf{Architecture sizes.} We consider two model sizes in this work, which we pretrain from random initialization:

1. RESERVE-B, with a hidden size of 768, a 12-layer ViT-B/16 image encoder, and a 12-layer joint encoder.
2. RESERVE-L, with a hidden size of 1024, a 24-layer ViT-L/16 image encoder, and a 24-layer joint encoder.

We always use a 12-layer audio encoder, and a 4-layer text span encoder. We pretrain on 192x320 images, and adapt the resolution to 288x512 widescreen images for downstream tasks. To simplify adapting to variable input sizes, we eschew learned position embeddings in favor of a multi-dimensional rotary position encoding [104, 10]. Details and hyperparameters are in Appendix B.

\section{3.2. Contrastive Span Training}

We introduce \textit{contrastive span} training, which enables learning across and between the three modalities. As shown in Figure 3, the model is given a sequence of video segments. For each one, we include the video frame, and then three \textquote{subsegments}, each that are either text or audio (never both, due to redundancy of text given audio). We train by replacing 25\% of these text and audio subsegments with a special \texttt{MASK} token. The model must match the representation atop the \texttt{MASK} \textit{only} with an independent encoding of its span.

Our approach combines past success at matching images to their captions [91, 62] along with \textquote{VisualBERT}-style prediction of independent tokens [106, 20] – though, crucially, we predict representations at a higher-level semantic unit than individual tokens. Our approach also enables the model to learn from both audio and text, while discouraging \textit{memorization} of raw perceptual input, or tokens – which can harm representation quality [112].

Formally, we minimize the cross entropy between the \texttt{MASKed} prediction \(\hat{w}_t\) and its corresponding phrase representation \(w_t\), versus others in the batch \(V\):

\[
L_{\text{mask} \rightarrow \text{text}} = \frac{1}{|V|} \sum_{w_t \in V} \left( \log \frac{\exp(\sigma \hat{w}_t \cdot w_t)}{\sum_{w \in V} \exp(\sigma \hat{w}_t \cdot w)} \right). \tag{1}
\]

We first \(L^2\)-normalize \(w\) and \(\hat{w}\), and scale their dot product with a parameter \(\sigma\) [91].\(^7\) We then add this to its transposed version \(L_{\text{text} \rightarrow \text{mask}}\), giving us our text-based loss \(L_{\text{text}}\). Analogously, we define \(L_{\text{audio}}\) for audio, between the \texttt{MASKed} prediction \(\hat{a}_t\) and its target \(a_t\), versus others \(a\) in the batch.

In addition to these masked text and audio objectives, we simultaneously train the model to match video frames with a contextualized encoding of the transcript. In MERLOT [126], this objective was found to be critical for learning visual recognition from self-supervised videos. Here, the joint encoder encodes the entire video’s transcript at once, extracting a single hidden representation per segment \(\hat{v}_t\). We use the same contrastive setup as Equation 1 to maximize the similarity of these vectors with the corresponding \(v_t\) vectors from the frames, giving us a symmetric frame-based loss \(L_{\text{frame}}\). The final loss is the sum of the component losses:

\[
L = L_{\text{text}} + L_{\text{audio}} + L_{\text{frame}}. \tag{2}
\]

\textbf{Avoiding shortcut learning.} Early on, we observed that training a model to predict a \textit{perceptual} modality (like audio or vision) given input from \textit{the same modality}, led to shortcut learning – a low training loss, but poor representations. We hypothesize that this setup encourages models to learn imperceptible features, like the exact model of the microphone, or the chromatic aberration of the camera lens [33]. We avoid this, while still using audio as a target, by simultaneously training on two kinds of masked videos:

\begin{enumerate}[i.]
\item \textbf{Audio only as target.} We provide only text and video frames. The model produces representations of both \textit{audio} and \textit{text} that fill in \texttt{MASKed} blanks. It thus learns from audio supervision, without intra-modality shortcuts.
\item \textbf{Audio as input.} We provide the model video frames, and audio (or subtitles) at each segment. Because the model is given audio as an input somewhere, the model only produces representations for \texttt{MASKed} \textit{text}.
\end{enumerate}

There is one more data issue that must be addressed to avoid shortcuts. YouTube’s captions are not perfectly time-aligned with the underlying audio. During our initial

\(^6\)The \texttt{CLS} hidden states are not involved in the spatial pooling, and likewise are not given to the joint encoder.

\(^7\)Following past work, we optimize \(\sigma\) and clip it at 100, which enables the model to \textquote{warm-up} its emphasis placed on hard negatives [91, 113].
exploration, models took ready advantage of this shortcut: for instance, predicting an audio span based on what adjacent (overlapping) words sound like. We introduce a masking algorithm to resolve this; details in Appendix C.

**Pretraining setup.** We train 🇺🇸 on TPU v3-512 accelerators; training takes 5 days for 🇺🇸, and 16 days for 🇺🇸. We made pretraining more efficient through several algorithmic and implementation improvements, listed in full in Appendix B.5. Of note, we provide the joint encoder at most $N=8$ frames at a time, and incorporate extra contrastive text negatives from written text.

We use a batch size of 1024 videos, each with $N=16$ segments. The transcript frame matching task uses all 16384 images from the batch. Likewise, the contrastive span task uses all 49152 audio clips and text candidates in the batch. We use AdamW [68, 79] to minimize Equation 2, details and hyperparameters are in Appendix B.5.

### 3.3. Pretraining Dataset

Recent prior work on static images that demonstrates empirical improvements by increasing dataset size – all the way up to JFT-3B [69, 34, 91, 128]. The same pattern emerges in videos: prior work that has shown promising empirical improvements not only by scaling to 6 million videos/180M frames [126], but also by collecting a diverse set (i.e., going beyond instructional videos [59]).

To this end, we introduce a new training dataset of 20 million English-subtitled YouTube videos, and 1 billion frames, called YT-Temporal-1B. At the same time, we take steps to protect user privacy, directing scraping towards public, large, and monetized channels. We detail our collection, preprocessing, and release strategy in Appendix E.

### 4. Experiments

In this section, we show that a finetuned 🇺🇸 obtains state-of-the-art results on both VCR [124] as well as TVQA [74], suggesting strong multimodal script knowledge understanding. We then show that our model has strong zero-shot capability, over four challenging zero-shot tasks.

#### 4.1. Visual Commonsense Reasoning

We evaluate 🇺🇸 first through finetuning on VCR [124], a challenging multiple choice task that is commonly used as a benchmark for vision-and-language models. Most competitive models for VCR learn exclusively from images paired with captions, often with supervised visual representations (like from an object detector). To the best of our knowledge, the only exception is MERLOT [126], which uses YouTube video frames and text as part of pretraining; no VCR model to date was pretrained on audio.

**VCR Task.** A model is given an image from a movie, and a question. The model must choose the correct answer given four multiple choice options ($Q \rightarrow A$); it then is given four rationales justifying the answer, and it must choose the correct one ($QA \rightarrow R$). The results are combined with a $Q \rightarrow AR$ metric, where a model must choose the right answer and then the right rationale, to get the question ‘correct.’

**Finetuning approach.** We finetune models using [126]’s approach: ‘drawing on’ VCR’s detection tags directly onto the image, and jointly finetuning on $Q \rightarrow A$ and $QA \rightarrow R$. For both subproblems, we consider four sequences (one per answer or rationale option). Each sequence has pooled image features, the question (and, in $QA \rightarrow R$, the answer); and then the option. We pool a hidden representation from a MASK inserted after the text. Using a newly-initialized $d_h$ vector, we extract a logit for that candidate, and use cross-entropy to optimize that score (details in Appendix D.1.1.)

#### 4.1.1 Ablations: contrastive learning with audio helps.

While we present our final, state-of-the-art VCR performance in 4.1.2, we first use the corpus for an ablation study, empirically justifying design decisions of our pretraining setup. We use the same architecture and data throughout, allowing apples-to-apples comparison between modeling decisions.8

**Contrastive Span helps for Vision+Text modeling.** We start by comparing finetuning objectives for learning from YouTube ASR and video alone:

- **a. Mask LM.** This objective trains a bidirectional model by having it independently predict masked-out tokens. We make this baseline as strong as possible by using SpanBERT-style masking [63], where text spans are masked out (identical to our contrastive spans).

  Each span $w$ is replaced by a MASK token. We predict each of its subwords $w_i$ independently: like [63], we concatenate the MASK’s hidden state with a position embedding for index $i$, pass the result through a two-layer MLP, and use tied embedding weights to predict $w_i$.

- **b. VirTex [27].** In this objective, we likewise mask text subsegments and extract their hidden states. The difference is that we sequentially predict tokens $w_i \in w$, using a left-to-right language model (LM) with the same architecture details as our proposed span encoder. The LM is given the pooled hidden state, and uses tied embeddings to predict new tokens $w_{i+1}$ conditioned on previous $w_{<i}$.

Results are in Table 1. Versus these widely used baselines, our contrastive span objective boosts performance by over 2%, after one epoch of finetuning only on vision and text. We hypothesize that its faster learning is caused by encouraging models to learn concept-level span representations; this might not happen when predicting tokens individually [22].

---

8We use the transcript-frame matching objective throughout, as MERLOT [126]’s ablations demonstrate that it is indispensable when training vision representations from random initialization. Our starting ‘Mask LM’ configuration is similar to MERLOT’s, and we improve from there.
Table 1: Ablation study of our contrastive span objective. It outperforms prior work even without audio; with an additional 1% boost when audio is used. Our full setup, adding written text, improves another 1%. A “*” denotes a component of our full model.

Table 2: RESERVE gets state-of-the-art leaderboard performance on VCR. We compare it with the largest submitted single models, including image-caption models that utilize heavy manual supervision (e.g. object detections and captions).

Audio pretraining helps, even for the audio-less VCR:

d. Audio as target. Here, the model is only given video frames and ASR text as input. In addition to performing contrastive-span pretraining over the missing text spans, it does the same over the (held-out) audio span (Equation 2). This boosts VCR accuracy by 0.7%.

e. Audio as input and target. The model does the above (for video+text input sequences), and simultaneously is given video+text+audio sequences, wherein it must predict missing text. This boosts accuracy by 1% in total.

f. Sans strict localization. We evaluate the importance of our strict localization in time. Here, in addition to correct subsegments at the true position $t$ as a correct match, we count adjacent masked out regions as well. An extreme version of this was proposed by [48], where a positive match can be of any two frames in a video. Yet even in our conservative implementation, performance drops slightly, suggesting localization helps.

Putting these all together, we find that contrastive span pretraining outperforms mask LM, with improved performance when audio is used both as input and target. For our flagship model, we report results in Table 1 on simultaneously training on web-text sequences as well (Appendix C.4), this improves performance by an additional 1%.

4.1.2 VCR Results

Encouraged by these results, we train our models for 10 epochs on YT-Temporal-1B. Figure 4 demonstrates that fine-tuned VCR performance tracks with the number of pretraining epochs, as well as the validation loss.9

Finally, in Table 2, we compare RESERVE against the largest published models from the VCR leaderboard. Of note, RESERVE-L outperforms all prior work, by over 5% on Q→AR metric. It outperforms even large ensembles (e.g. 15 ERNIE-Large’s) submitted by industry [123], though we do not show these on this table to focus on only single models.

Efficiency. The accuracy increase of RESERVE is not simply due to compute.10 In fact, our RESERVE-L requires one-fifth the FLOPs of detector-based systems, like UNITER-Large [20] (Appendix B.3). Moreover, because RESERVE-L uses a pure ViT backbone versus MERLOT’s ViT-ResNet hybrid, it uses fewer FLOPs than MERLOT, while scoring 7% higher. Meanwhile, RESERVE-B outperforms ‘base’ detector-based models, while using less than one-tenth their FLOPs.

In terms of parameter count, RESERVE-B is comparable to prior work. On VCR, including the vision stack, RESERVE-B has 200M finetunable parameters and performs similarly to the 378M parameter UNITER-Large. RESERVE-L has 644M parameters.

4.2. Finetuning on TVQA

We use TVQA [74] to evaluate our model’s capacity to transfer to video understanding tasks when finetuned. In TVQA, models are given a video, a question, and five answer choices. The scenes come from American TV shows, and depict characters interacting with each other through dialogue – which past work represents through subtitles. We will show that providing audio to RESERVE improves performance.

Audio-Subtitle Finetuning. To evaluate how much audio can help for TVQA, we finetune RESERVE jointly between

---

9The plot suggests that if we pretrained longer, VCR performance might continue to increase. While this could be true, an additional confounding factor is the learning rate decay schedule. With access to compute beyond our current capacity, future work would be well-suited to consider this and other pre-training modifications.

10Here, we use FLOPs as our key efficiency metric, as they are a critical bottleneck in model scaling [65, 34, 128]. On the other hand, we argue that parameter count can be misleading – for instance, many Transformer parameters can be tied together with minimal performance loss [71].
4.3. Zero-Shot Experiments

Next, we show that our model exhibits strong zero-shot performance for a variety of downstream tasks. Our zero-shot interface is enabled by our contrastive span objective. For QA tasks that require predicting an option from a label space of short phrases, we encode this label space as vectors, and predict the closest phrase to a masked input. We consider:

i. Situated Reasoning (STAR) [119]. This task requires the model to reason over short situations in videos, covering four axes: interaction, sequence, prediction, and feasibility. The model is given a video, a templated question, and 4 answer choices. We convert templated questions into literal statements (which are more similar to YouTube dialogue); the label space is the set of four options.

ii. Action Anticipation in Epic Kitchens [26]. Here, the goal is to predict future actions given a video clip, which requires reasoning temporally over an actor’s motivations and intentions. The dataset has a long tail of rare action combinations, making zero-shot inference challenging (since we do not assume access to this prior). As such, prior work [45, 38] trains on the provided in-domain training set. To adapt RESERVE to this task, we provide it a single masked token as input text, and use as our label space of all combinations of verbs and nouns in the vocabulary (e.g., ‘cook apple, cook avocado’, etc.).

iii. LSMDC [81, 95]. Models are given a video clip, along with a video description (with a mask to be filled in). We compare it with the vocabulary used in prior work [126].

iv. MSR-VTT QA [120]. This is an open-ended video QA task about what is literally happening in a web video. We use GPT3 [16], prompted with a dozen (unlabelled) questions, to reword the questions into statements with masks. This introduces some errors, but minimizes domain shift. We use [122]’s label space, with 1k options. For these tasks, we use N=8 video segments (dilating time when appropriate), and provide audio input when possible. Details and prompts are in Appendix D. We compare against both finetuned and zero-shot models, including running CLIP [91] on all tasks. CLIP is a strong model for zero-shot classification, particularly when encyclopedic knowledge about images is helpful; our comparisons showcase where multimodal script knowledge helps.

Results. Table 4 shows our model performs competitively:

i. On STAR, RESERVE obtains state-of-the-art results, outperforming finetuned video models. It performs well on EPIC-Kitchens (verb and noun forecasting), along with LSMDC, despite their long-tail distributions. On MSR-VTT QA, it outperforms past work on weakly-supervised video QA. Further, it outperforms CLIP (that cannot handle dynamic situations), and benefits from audio when given.

Table 3: RESERVE gets state-of-the-art results on TVQA by a margin of over 7%, over methods from prior work – that do not make use of audio information. The ‘vision and subtitles’ and ‘vision and audio’ settings. Like on VCR, we consider one sequence per candidate: each contains video frame features, the question, the answer candidate, and a Mask token (from where we pool a hidden representation). During training, each sequence is duplicated: we provide one sequence with subtitles from the video, and for the other, we use audio. This lets us train a single model, and then test how it will do given subtitles, given audio, or given both (by averaging the two softmax predictions).

Table 4: Zero shot results. RESERVE obtains state-of-the-art results, outperforming finetuned video models. It performs well on EPIC-Kitchens (verb and noun forecasting), along with LSMDC, despite their long-tail distributions. On MSR-VTT QA, it outperforms past work on weakly-supervised video QA. Further, it outperforms CLIP (that cannot handle dynamic situations), and benefits from audio when given.

<table>
<thead>
<tr>
<th>Model</th>
<th>TVQA (acc.%)</th>
<th>Situated Reasoning (STAR)</th>
<th>EPIC-Kitchens</th>
<th>LSMDC</th>
<th>MSR-VTT QA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Val Test</td>
<td>Situated Reasoning</td>
<td>EPIC-Kitchens</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Human [74]</td>
<td>-- 89.4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MERLOT [126]</td>
<td>78.7 78.4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MMPI-BERT [109]</td>
<td>73.5 72.8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Kim et al [67]</td>
<td>76.2 76.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RESERVE-B (subtitles)</td>
<td>81.3 –</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RESERVE-L (subtitles)</td>
<td>85.6 84.8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RESERVE-B (audio)</td>
<td>82.5 –</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RESERVE-L (audio)</td>
<td>85.9 85.6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RESERVE-B</td>
<td>83.1 82.7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RESERVE-L</td>
<td>86.5 86.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Results. We show TVQA results in Table 3. Without audio so as to enable fair comparison, our RESERVE-B outperforms all prior work with a margin of 2.6%. Audio performs even better than subtitles. And, combining subtitle-only and audio-only predictions performs best of all, improving over 4% versus the prior state-of-the-art, MERLOT (and in turn over other models). The same pattern holds (with additional performance gains) as model size increases: RESERVE-L improves over prior work by 7.6%.
heavy-tailed nature of both distributions. It is worse on getting both right (‘action’), we suspect that this might be due to priors (motifs) between noun and verb [127]. These are easy to learn given access to training data, but we exclude these as we consider the zero-shot task.

iii. On LSMDC, our model obtains strong results at filling-in-the-blank, likewise despite a heavy (unseen) frequency bias. Notably, it outperforms CLIP significantly, with CLIP often preferring templates that use visually-relevant words, even if they don’t make sense as a whole.11

iv. Finally, our model performs well on MSR-VTT QA, outperforming past work that directly rewords subtitled instructional videos into video QA instances [122].

5. Qualitative Analysis: Why does audio help?

What can RESERVE learn from both text and audio? Three validation set examples are shown in Figure 5. The model is given the displayed text and video frames, and must match the MASK to the correct missing text and audio span (out of 48k total in the batch). The plots show RESERVE-B’s probability of correctly identifying the correct audio or text span, as it progresses through 10 epochs of pretraining.

Audio’s supervisory signal. In the first two rows of Figure 5, audio provides orthogonal supervision to text:

1. The first row is from the recurring ‘popcorn’ example, and the MASKed audio contains the distinctive sound of popcorn pops slowing, as most kernels have already finished. By the final epoch, RESERVE-B selects this specific auditory cue with 60% probability, over others (including from adjacent frames, with different distinct stages of popping). Here, sound provides signal for learning joint vision-text understanding of the situation. This cannot be captured by predicting the MASKed text ‘and forth every now and then,’ as evidenced by the lower match-probability.

2. The second row contains only the text ‘why,’ with the audio providing greatly more information — a female-presenting speaker (shown in the next frame) laughs, astonished that the child (in the frame afterwards) might want a better relationship with their parents.

3. In the third row, matching performance is similar between modalities, possibly as the yogi is narrating over a (muted) video recording, and not adding much information.

Role of text. Text is still a crucial complement to audio, in terms of the supervision it provides. Consider the second row: RESERVE-B learns to match the audio almost perfectly (perhaps reasoning that the speaker is shown in the next frame, and is laughing). In later epochs, its text-match probability increases: knowing that a ‘why’ question is likely to be asked is a valid social inference to make about this (tense) situation.

Learning through multimodal reentry. In this work, we proposed a reentrant learning paradigm: predicting a modality like audio given fused input from vision and text. This paradigm contrasts with today’s standard practice of learning primarily through language (either from language alone, or from images paired with captions).

Indeed, we argue that learning from audio through reentry goes beyond offering orthogonal signal to models. Our pretraining task forces RESERVE to abstract away from intramodal perceptual features (like pixels or tokens), towards learning a more complete understanding of the world.

---

11For instance, on a picture of a smiling mailman ready to leave, CLIP chooses ‘the mailman smiles off,’ versus ‘the mailman takes off.’
as viewed from the held-out modality. Developmental psychologists have hypothesized that human children adopt such a learning strategy, learning connections not just between vision, language, and audio but also touch, smell, and taste [35, 100]. Our work suggests it has potential for machines as well.

6. Conclusion, Limitations, Broader Impact

We introduced RESERVE, which learns representations jointly through sound, language, and vision, guided through a new pretraining objective. Our model performs well in both finetuned and zero-shot settings, yet there are still limitations:

a. Our model only learns to reason over 40-second long video chunks at once (8 ‘segments’).

b. Our setup learns only from videos with English ASR subtitles. It would not work for the majority of the world’s languages, where ASR systems today do worse.

c. Our model can only match, not generate, text and audio.

Still, we foresee several potential broader societal impacts of our work. An expanded version of this technology might someday serve as a component of assistive technology for improved video captioning, e.g., to improve accessibility for low vision and blind users [75] or for people who are d/Deaf [47]. Yet, the same technology can have impacts that we authors consider to be negative, including surveillance, or applications that hegemonize social biases that are widespread on YouTube. We discuss these further in Appendix A: key dimensions include respecting user privacy during dataset collection, exploring biases in YouTube data, dual use, and energy consumption. We discuss our plan to release our model and data for research use so others can critically study this approach to learning script knowledge.

Acknowledgements

We thank Jae Sung Park, Oren Etzioni, Gabriel Ilharco, and Mitchell Wortsman for feedback on this work. Thanks also to Zak Stone and the Google Cloud TPU team for providing access to the TPU machines used for conducting experiments. Thanks to James Bradbury and Skye Wanderman-Milne for help with JAX on TPUs. Thanks to the AI2 ReViZ team, including Jon Borchardt and M Kusold, for help with the demo. This work was funded by DARPA MCS program through NIWC Pacific (N66001-19-2-4031), and the Allen Institute for AI. Last, but not least, thanks to the YouTubers whose work and creativity helps machines to learn about the multimodal world.

References


[22] Kyunghyun Cho. Tweet, "important dependences between the image features and words/phrases in the description could be explained away by the dependencies among words/phrases". 5


[38] Antonino Furnari and Giovanni Maria Farinella. Rolling-unrolling lstms for action anticipation from first-person video. IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), 2020. 7, 25, 26


[88] Jean Piaget and Margaret Trans Cook. The origins of intelligence in children. 1952. 1


[114] Luyu Wang, Pauline Luc, Adria Recasens, Jean-Baptiste Alayrac, and Aaron van den Oord. Multimodal self-
Conference on Computer Vision and Pattern Recognition, 2018. 8

[128] Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers, 2021. 5, 6, 18


Abstract

We provide the following materials in the appendix:
• A full broader impact statement (Section A)
• Details about our model architecture (Section B)
• Details about how we provide video data into the model, including how we align the modalities and perform the masking (Section C)
• Details about how we adapted our model to downstream tasks (Section D)
• Details about how we collected data (Section E)
• Additional experiments (Section F)

A. Broader Impact Statement

In this paper, we have presented a model for learning multimodal neural script knowledge, through incorporation of audio as a first-class citizen alongside text and video frames. We argue that academic study of this learning paradigm is important, in part because it relates to how we as humans understand the world. We as humans process situations by perceiving through multiple modalities and interpreting the result holistically.

At the same time, the work and methodology that we outlined risks dual use. Like other large machine learning systems pretrained on web data, our system may reproduce harmful social biases present in its training data. While a variety of past work has studied risks of language-only pretraining [125, 14, 8, 60], the video-centric pretraining that we explore in our work might have different benefits and risks. We discuss these below, along with how we worked to mitigate them through our work.

A.1. Privacy.

A significant risk with training on data at YouTube scale is protecting user privacy. We took several proactive steps to ensure this, that in turn build off prior work and community norms [1, 83, 126]:

a. We plan to release only video IDs for download, following prior work [1, 83]. Thus, when a user deletes a video off of YouTube, it becomes removed from YT-Temporal-1B as well, giving content creators a right to opt out of all uses of their videos.

b. Building off of past work [126], we directed our data collection towards public and monetized channels. These channels are identifiable insofar as they contain more subscribers, and more videos. They include companies that have official accounts, including journalism outlets like the New York Times and Vox. They also include individuals for whom making public YouTube videos is their full time job. In either case, our use videos in question for research purposes can be seen as fair use.

Framing of privacy. Privacy is a nuanced topic with many societally, culturally, and generationally-specific interpretations. We took inspiration from Marwick and Boyd [82]’s framework of networked privacy, which posits that users posting public videos might encode private information – enough so that their intended viewership (friends, possibly) can catch the gist, but not enough so as to leak private details like phone numbers to the world.

Through the lens of networked privacy, we see key differences between studying videos on a moderated platform, versus NLP work that trains models from the open web (e.g. [92, 16]). When YouTube users upload videos, they seem to understand details of its privacy policy, beyond consenting to it [64]. YouTubers typically upload their own videos [102]. These details differ from text on the open web. Today, ‘data brokers’ post private details (like phone numbers) to the web for profit [24]; concerningly, a study on language models suggests that models are vulnerable at memorizing this private information [18].

It is worth examining our research through other framings of privacy as well. For example, internet platforms profit off of user data, whereas users do not share equally in these profits [37]. For this, and for the other reasons mentioned, we aim to release our model only for research-based use.

A.1.1 Empirical study: can RESERVE identify individual celebrities?

Inspired by work studying language model memorization of private information [18], we wish to empirically probe RESERVE’s ability to recognize individuals. Our goal during model development was not to optimize for this ability. Instead, our goal was to study models for multimodal script knowledge (what people might be doing in a situation over time, and why) instead of long-tailed visual recognition (including who those individuals are). These goals might trade off – for instance, our training data only has individuals’

<table>
<thead>
<tr>
<th>Model</th>
<th>Voice</th>
<th>Image+Voice</th>
<th>Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESERVE-L</td>
<td>10.8</td>
<td>9.6</td>
<td>10.7</td>
</tr>
<tr>
<td>CLIP ViT-B/16 [91]</td>
<td></td>
<td></td>
<td>86.0</td>
</tr>
</tbody>
</table>

Table 5: Zero-shot person (face/voice) recognition accuracy on VoxCeleb2 [86] and VGGFace2 [17], using different modalities. While RESERVE can perform person recognition from several modalities, its performance is much lower than the recognition-optimized CLIP model in the image-to-name setting. We hypothesize that this is due to a similarity between this setting and CLIP’s pretraining data – news articles often include celebrity images, paired with their names.
names when they are mentioned in ASR subtitles, a pairing that might be significantly noisier than images and text on the open web.

We study this capacity on the VoxCeleb2 and VGGFace2 datasets [86, 17], where we created a test set of 120 celebrities, with 100 samples of each. We study these datasets not to promote them, but to establish a conservative upper-bound for the capacity of a model to recognize non-celebrities. We hypothesize that if REFRESERV struggles to select the right celebrity out of 120 predefined random options, it would struggle much more at identifying random people (where the set of candidate names is much greater). We test this hypothesis over three zero-shot settings:

1. **Voice to name.** Given an audio clip sampled for a celebrity, we encode it with our model’s audio encoder. We provide our model’s joint encoder the text ‘the sound of MASK’, followed by the encoded audio. A blank image is provided. We extract the representation on top of the MASK, and choose the most similar celebrity name.

2. **Image+voice to name.** Here, we adopt the same format as ‘Audio to name,’ except we additionally encode an image of the celebrity’s face in question.

3. **Image to name.** Here, REFRESERVE encodes an image of the celebrity in question, and we provide it with text ‘A picture of MASK’. No audio is provided. Using our model’s joint encoder, we select the closest encoded celebrity name, out of all options.

We use this format to compare to a CLIP model, which was trained on web images with captions [91]. For the CLIP comparison, we use it to encode each image, and for all considered celebrity names, the sentence ‘A picture of ${name}’ . We choose the closest encoded sentence to the encoded image.

We show our results in Table 5. In all modes, our model is less than 11% accurate at recognizing celebrities. Curiously, the accuracy drops given both the image and the voice, suggesting that the way we fused a celebrity’s image and voice together might be outside the model’s training distribution. These results are significantly lower than CLIP’s 86% accuracy at classifying a person from their image.

In Figure 6, we investigate more into which celebrities our model is best at recognizing. Only a few celebrities are reliably classified; these tend to be very famous celebrities like Oprah Winfrey and Justin Bieber. Several sports players are recognized well (including Lebron James and Roger Federer), which could imply that our model learned their identities from watching sports replays or commentary. Most other celebrities are hardly recognized, whereas CLIP does well across the board.

**Results summary.** Together, these results show that while models like CLIP focus on encyclopedic knowledge that results in strong zero-shot person recognition accuracy,
RESERVE is not as effective as other models in memorizing particular celebrities – and, thus, perhaps not as effective as memorizing particular non-celebrities. These results suggest that RESERVE’s objectives and data might make it less of a concern to release privacy-wise, versus models trained on web images with captions.

As the rest of the paper emphasizes however, RESERVE performs well on tasks with temporal understanding and commonsense reasoning as the primary goal. On a broader level, these results suggest that it is possible to learn strong models about temporal reasoning without person-level memorization, though more work is needed.

### A.2. Biases in (pre)training data.

The ‘knowledge’ that our model learns should be viewed as situated within YouTube [66], which has numerous biases (that we will discuss next). Past work has made similar observations for language model pretraining on the open web[8]. One of the root causes of such bias is learning objectives that encourage memorization of surface level cooccurrences, rather than truly causal factors [55, 9, 117]. Though it is possible that in the very long term, a paradigm of grounded learning might resolve some of these issues, the objectives in this work still likely reify biases that exist in the YouTube data.

**Platform biases.** Unlike many other pretraining efforts, that scrape data from the open internet (e.g. [92, 16, 91]) which directly leads to toxic biases (e.g. [41, 32, 11]); we trained our model on YouTube, which is a moderated platform [101]. Though the content moderation might perhaps reduce overtly ‘toxic’ content, social media platforms like YouTube still contain harmful microaggressions [15], and alt-lite to alt-right content [94]. Additionally, it should be mentioned that the content moderation on YouTube disproportionately filters out minoritized voices [43]. Thus, despite us not using any word-based ‘blocklist,’ our model’s pretraining data is still biased [32]. Even without videos being explicitly removed, the ‘YouTube algorithm’ incentivizes the production of certain types of content over others [12, 102]; e.g. people’s roles in YouTube videos tend to be highly gendered [85], which might bias situation understanding [131].

**Bias amplification.** In this work, we pretrained a model primarily on ASR text, which is itself produced by another model. The automatic captions in YouTube are known to suffer from gender bias [107], which our model (like neural models generally) might in turn amplify [131]. The transcriptions on YouTube are also likely poor at handling important identity markers, like pronouns. Already, text-only models like BERT struggle with pronouns like they/them and z/zi/zer; our reliance on ASR text makes our corpus likely worse in this regard [28]. While past work, namely MERLOT [126], ‘cleaned’ this ASR text – through another large language model – we opted not to do so for this work due to computational expense. Though in that work, the ASR-denoiﬁcation was found to boost performance in VCR, it seems unlikely that it would solve this core issue of model bias.

### A.3. Dual use.

Learning connections between video, audio, and text – though an important area of study as we have argued – can be used for undesirable applications, beyond what we have outlined under ‘bias.’ We outline and discuss a few below.

**Generating fake content.** A concern for pretrained models is that they can generate fake content, that could be used by ‘bad’ actors for their ends [125]. It should be noted that our model cannot explicitly ‘generate’ text, audio, or vision in a direct sense. Nonetheless, however, it is possible that a ﬁnetuned or expanded version of this model could be used for that purpose – and that our model would be more helpful to such an actor versus them training their own (perhaps larger) model from scratch.

**Surveillance.** Our model might contain representations that enable it to be used in surveillance applications. As we note in Appendix A.1.1, our model’s low performance on person recognition suggests that it might perform poorly recognition-focused applications. Still, one possibility is that a neural script knowledge could ‘summarize’ surveillance videos in some form (like identifying an activity of interest), without identifying the person(s).

We suspect (but cannot deﬁnitively prove) that the reporting bias of the YouTube data that it was trained on might make it poor for such a surveillance-focused task [49]. Namely, most surveillance videos are sparse in nature – finding an activity of interest is like finding a ‘needle in a haystack’ [90]. Though, some surveillance videos are inevitably posted on YouTube and then captioned, these disproportionately contain interesting events (like somebody’s car crashing into a house). It is not clear whether our system could be easily adapted to such a sparse problem; the amount of work required suggests that it might be out-of-scope at least for low-skill actors. On the other hand, this broad research agenda, and perhaps all of computer vision for that matter, might enable large actors to do just that [132]; which might not be addressable through purely technical solutions [51].

**Harmful outcomes if deployed.** Beyond the biases that our system possesses, some applications of our system – if deployed in production – could cause harm, particularly to groups already harmed by AI systems. Of note, linking someone’s voice with their appearance is not always a good thing [93]. Likely some of the key features that our model learns – though we did not teach it this explicitly – involve recognizing gender, and this is harmful especially to transgender individuals [54].

Our model cost a lot amount of energy to pretrain \([103]\); roughly 3 weeks of time on a TPU v3-512. The total carbon footprint of our work was a net 8.23 tons of CO\(_2\) equivalent, which is roughly 4.5% of the emissions of a jet plane flying round-trip from San Francisco to New York.\(^{12}\)

At the same time, it is possible that our model could save energy overall, when shared with researchers who build off of our system. Indeed, \(\text{RESERVE-B}\) uses less energy than MERLOT \([126]\) (due to a smaller vision backbone, and smaller image sizes), MERLOT in turn is more efficient than past work which used expensive detector-based backbones (e.g. \([106, 20, 129]\)), that are made more expensive because some of their computational primitives (like non-maximum suppression) are difficult to make efficient on-device.

A.5. Synthesis.

With these risks in mind, we release our video IDs, as well as \(\text{RESERVE’s}\) checkpoints, exclusively for research use. We believe that at this point in time, we as a field lack full knowledge of the privacy, bias, and dual-use risks of video-based models – though, we hope that our analysis in this section provides a good starting point. For instance, while the objectives that we have studied were designed to promote learning general neural script knowledge above *encyclopedic memorization*, they have not yet been tested in all possible cases. By opening our models to the research community, we hope to promote fundamental work in uncovering both promising aspects of these systems, alongside examining their risks. We hope to contribute to these lines of research as well.

B. Model implementation details

In this section, we discuss at a more in-depth, technical level, how we implement certain aspects of \(\text{RESERVE,}\) and other details (like its runtime in FLOPs).

B.1. Rotary position encoding

As mentioned in the main text, we use a rotary position encoding to model the relative location of input sequences \([104, 10]\). We chose this primarily because we did not want to use absolute (additive) position embeddings, which would have to be added to the inputs of each encoder, and possibly at multiple levels in the hierarchy (e.g. for the joint encoder, the video segment index \(t\) would be needed as well).

The rotary encoding uses no parameters, and instead uses a kernel trick to allow the model to recover relative distances between key and query elements in a Transformer’s attention head. This can be seen as ‘rotating’ pairs of elements; we apply the rotation to only the first half of each 64-dimensional head, and the second half is kept as is.

**Multidimensional coordinates.** We treat each token as having a 4-dimensional position of \((h, w, \ell, t)\), corresponding to the \(h, w\) coordinates in the image, the position \(\ell\) in the text-sequence, and the segment index \(t\). If a dimension is irrelevant to a modality (like \(h, w\) for text), we set it to 0. Thus, for our various encoders, we use the following coordinate schemes:

a. Video Frame Encoder (ViT): just the \(h, w\) coordinates of the image; so \((h, w, 0, 0)\).

b. Audio Encoder: Only the 1-D position \(\ell\) of the patch in the spectrogram: \((0, 0, \ell, 0)\).

c. Text Span Encoder: Only the 1-D position \(\ell\) of the token in the input: \((0, 0, \ell, 0)\).

d. Joint encoder: Here, we use all coordinates. Inputs from the video frame encoder have coordinates \((h, w, 0, t)\), where \(t\) is their segment index. The text and (pooled) audio inputs are merged, and they each have coordinates \((0, 0, \ell, t)\), where \(\ell\) here is the absolute position in the entire sequence (across segments).

As part of our implementation, we normalize the rotary coordinates. \(h, w\) are scaled to be in the range \([-1/2, 1/2]\), such that text is implicitly ‘in the center’ of the image. Likewise, \(\ell\) and \(t\) are scaled to be in the range \([0, 1]\). The positions are used to compute relative distances, by using a kernel trick to rotate coordinates in the keys and values of each \(d_h\)-sized Transformer attention head.

B.2. Sequence lengths

We briefly remark on the sequence lengths used by parts of the model.

a. Video Frame Encoder (ViT): Most YouTube videos are widescreen (16x9). We thus used a widescreen resolution for our video frame encoder. It takes in patches of size 16x16, and we used a layout of 12 patches (in height) by 20 patches (in width). This corresponds to \(192\times320\). Among other factors that are important are ensuring that TPUs do not excessively pad the sequence length \([128]\). The sequence length is 241 in this case, as there is a CLS token, and it gets padded to 256.

As we note in the main text, afterwards we apply attention pooling in a 2x2 grid (ignoring the CLS token here), which gives us a \(H/32\) by \(W/32\) grid for the joint encoder (6 x 10).

\(^{12}\)CO\(_2\) Calculation. It is also important to consider the location where these TPUs are located, as the renewables portion at each datacenter is not equal \([87]\). Our TPUs were in the ‘Europe-west4’ region, which uses on average 90% carbon-free energy, and a Grid Carbon intensity of 0.410 kgCO\(_2\)/eq kWh. A single TPU v3 processor (with 8 cores over 2 chips) has a power average of 283W, so after performing the math from \([87]\), our training cost 20,000 kWh. This gives us a net 8.23 tons of CO\(_2\) equivalent. It should be mentioned that this figure only covers the electricity usage given the chips (and the datacenter), not the raw materials involved in making these chips (which is significant \([111]\)].
b. Audio Encoder. Our model independently encodes each 1.6 second of audio. We do this through spectrograms. Each window involves 1536 samples at a sample rate of 22500 Hz, and there are 588 samples ‘hops’ between windows. We chose these hyperparameters largely around efficiency. We found that the Discrete Fourier Transform is fastest if the window size is close to a multiple of 2. We used a small number of mel spectrogram bins (64) because we found that at that threshold, we could reconstruct the original sequence at an acceptable level using the Griffin-Lim algorithm, [52] which itself might be a lower bound on quality as neural methods trained for this purpose have been shown to do better [115].

In our implementation, we compute the spectrogram for an entire video segment (5 seconds) at once; this is of size 64 mel bins by 192 windows. During pretraining, we perform what is effectively a ‘random crop’ over the spectrogram: we extract three sequential 64x60 sub-spectrograms, for each audio subsegment. We constrain them to not overlap, which means that 12 (random) windows are held out.

We note that our Audio Encoder AST is quite different from the one proposed by [46]. Though it operates over spectrograms, we opted for a linear ‘1-dimensional’ layout rather than a two-dimensional (image-like) one. We also did not pretrain our audio encoder on any supervised data (they used ImageNet and found, perhaps surprisingly, that it helped initialize the model). We used a patch size of 64 mel bins by 2 windows; the resulting (1D) sequence is of size 30. After adding a CLS token, the result is a sequence of length 31.

As we note in the main text, we apply attention pooling afterwards (for all elements except the CLS token), pooling by a factor of five to resize the length-30 sequence to a length of 6 ‘audio tokens.’

c. Text Span Encoder: We operate on spans that are at most of length 15, with an additional CLS token. Its length is thus 16.

d. Joint encoder. Let L be the number of text or pooled audio tokens given to the model per segment, on average; we set $L=20$. Let T be the number of video segments. Then, the joint model’s sequence length is $T \times (L + W/32 \times H/32)$. Our total sequence length was thus 640.

To better adapt our model to downstream tasks – particularly single-image tasks like VCR [124], where past work tends to use a resolution much higher than 192x320, after pretraining, we performed FixRes pretraining (for one epoch on \texttt{RESERVE-B}, and one half epoch on \texttt{RESERVE-L} [108].

Here, we trained the model on larger images – simultaneously on 288x512 widescreen images (18 patches by 32 patches), and on 384x384 square images (24 patches on each side). The joint encoder, correspondingly, uses a sequence length of 1312.

During 10 epochs of pretraining, we used a cosine decay of the learning rate down to 0.02 its maximum. During FixRes pretraining afterwards, we warmed up the learning rate to 0.02x its peak, over the first 1/5th of an epoch, and afterwards used a cosine schedule to anneal it towards 0.

**B.3. Efficiency metrics of our model**

In Table 6, we report efficiency metrics of \texttt{RESERVE}, versus others. We calculate these metrics in the context of scoring a single VCR question and answer candidate. This requires encoding one image, and using 128 tokens for each question and answer combined (for all models). We compare against a UNITER [20], which is a representative Visual-BERT style model, along with MERLOT [126]. Our models are far more efficient in terms of FLOPs, with \texttt{RESERVE-L} being roughly on par with MERLOT, yet outperforming it by 6% in terms of VCR accuracy. We discuss key differences below:

<table>
<thead>
<tr>
<th>Model</th>
<th>GFlops, from</th>
<th>VCR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Image</td>
<td>Joint Encoder</td>
</tr>
<tr>
<td>UNITER-Base[20]</td>
<td>1766</td>
<td>28</td>
</tr>
<tr>
<td>UNITER-Large[20]</td>
<td>1767</td>
<td>99</td>
</tr>
<tr>
<td>MERLOT [126]</td>
<td>236</td>
<td>67</td>
</tr>
<tr>
<td>\texttt{RESERVE-B}</td>
<td>99</td>
<td>46</td>
</tr>
<tr>
<td>\texttt{RESERVE-L}</td>
<td>176</td>
<td>165</td>
</tr>
</tbody>
</table>

Table 6: Efficiency metrics of our model versus others, measured in terms of (giga) floating point operations required to process a single image, question, and answer candidate on VCR. We compare with the overall VCR performance on the combined Q→AR metric. Our \texttt{RESERVE} family of models are significantly more efficient than prior work, with \texttt{RESERVE-L} being roughly on par with MERLOT [126] in terms of FLOPs, yet improving accuracy by over 6%.

by 32 patches, and on 384x384 square images (24 patches on each side). The joint encoder, correspondingly, uses a sequence length of 1312.

During 10 epochs of pretraining, we used a cosine decay of the learning rate down to 0.02 its maximum. During FixRes pretraining afterwards, we warmed up the learning rate to 0.02x its peak, over the first 1/5th of an epoch, and afterwards used a cosine schedule to anneal it towards 0.

**B.3. Efficiency metrics of our model**

In Table 6, we report efficiency metrics of \texttt{RESERVE}, versus others. We calculate these metrics in the context of scoring a single VCR question and answer candidate. This requires encoding one image, and using 128 tokens for each question and answer combined (for all models). We compare against a UNITER [20], which is a representative Visual-BERT style model, along with MERLOT [126]. Our models are far more efficient in terms of FLOPs, with \texttt{RESERVE-L} being roughly on par with MERLOT, yet outperforming it by 6% in terms of VCR accuracy. We discuss key differences below:

a. UNITER. We note that UNITER, like other VisualBERT models, uses a supervised object detection backbone [5]. This processes images using a ResNet 101 model [56], at a resolution of 600x800; the final ResNet ‘C4’ block is applied densely over the entire image to obtain object-detection potentials everywhere in the image. Both factors greatly increase the FLOPs count.

When computing UNITER’s FLOPs count, we exclude operations like non-max suppression, which is an operation that is difficult to implement (and thus whose FLOP count might vary significantly depending on implementation). Our FLOPs count is thus a lower-bound. 36
detection regions are extracted, which is why the ‘joint encoder’ for UNITER is smaller than the equivalents for MERLOT and RESERVE.

b. MERLOT. This model has two key differences versus our RESERVE. First, it uses a larger image resolution for VCR: 384x704, versus our 288x512. Second, it uses a hybrid ViT-ResNet50 backbone for encoding images. The backbone here is lighter weight than the object detection backbone of UNITER (in particular, the final ‘C4’ block is removed), and thus, as shown in Table 6, though it uses more FLOPs than does our RESERVE-L, it uses far fewer FLOPs than UNITER.

We choose FLOPs as our primary comparison metric as past work shows that it is one of the key factors in model scaling [65, 34]. Parameters are arguably more fungible. For instance, in text-only representation learning, ALBERT [71] demonstrates that it is possible to tie parameters together at all layers of a BERT-like transformer, reducing parameters by an order of magnitude (while not modifying compute), with a minimal performance drop. We did not do this for this work, as we wanted to use a more ‘vanilla’ Transformer architecture; however, it suggests that representation learning models with hundreds of millions of parameters might be FLOPs bound as opposed to parameter-bound.

Nonetheless, UNITER-Base has 154 million parameters, though some are frozen (86 million from their Transformer; 23 million from the word embedding layer, and then 44 million from their object detector [5]). UNITER-Large has 378 million parameters (303 from their Transformer, 31 million from word embeddings, and 44 million from the same object detector. Meanwhile, MERLOT has 223M parameters. Versus our RESERVE-B, 14 million extra parameters are due to a larger vocabulary, and 10 million parameters are due to a ResNet50 encoder – but these parameters have a disproportionate impact in FLOPs count.

**B.4. Full model hyperparameters**

In Table 7, we present full hyperparameters for our model. Among other details, we used AdamW as our optimizer, with $\beta_2 = 0.98$ and $\epsilon = 1e-6$. We increased the learning rate linearly to its peak value (4e-4 for RESERVE-B, 3e-4 for RESERVE-L) over 3750 steps ($\frac{1}{20}$th of an epoch). Our number of warmup steps is lower than many other pretraining work; we note that all of our contrastive objectives involve learning a $\sigma$ parameter, which functions as a secondary ‘warmup.’

We did not use gradient clipping. We trained and evaluated in 16-bit bfloat16 precision wherever we could – casting all gradients to that precision as well, and saving the AdamW running mean and variance to be 16-bit as well. A few times during pretraining RESERVE-L, we found that some values in gradients would be NaN. We addressed this by always setting NaN values to be 0. This seemed to address

|                           | Base                  | Large
|---------------------------|-----------------------|-----------------------
| Sample rate               | 22050 Hz              |                       |
| Audio size                | 588 samples           | 1536                  |
| FFT hop length            |                       |                       |
| FFT window size           |                       |                       |
| Mel bins                  |                       | 64                    |
| Subsegment length         |                       | 60 hops, (~1.6 sec)   |
| Patch size                |                       | 64 mels x 2 hops      |
| Pooling ratio             |                       | 5                     |
| Final size                |                       | 6 tokens              |
| ViT patch size            | 16                    |                       |
| Pretraining size          | 192 x 320             |                       |
| Res-adaptation size       | 288x512 and 384x384   |                       |
| Pooling window            | 2 x 2                 |                       |
| Max. span length          | 15                    |                       |
| Mean span length          | 5.5                   |                       |
| N video segments          | 16                    |                       |
| video segment groups      | 2 (each with 8 segments) |                       |
| Pretraining seq. length   | 640 (160 text&pooled audio; 480 pooled vision) |                       |
| Res-adapted seq. length   | 1312 (160 text&pooled audio; 1152 pooled vision) |                       |
| Videos                    | 1024                  |                       |
| # Frames (for matching)   | 16384                 |                       |
| Masking rate              | 25% (of subsegments)  |                       |
| Text spans                | 49152                 |                       |
| Audio spans               | 49152                 |                       |
| Hidden size               | 768                   | 1024                  |
| Num attention heads       | 12                    | 16                    |
| Size per head             | 64                    |                       |
| Rotary size (per head)    | 32                    |                       |
| Vision num layers         | 12                    | 24                    |
| Audio num layers          | 12                    |                       |
| Text-span num layers      | 4                     |                       |
| Joint num layers          | 12                    | 24                    |
| Peak learning rate        | 4e-4                  | 3e-4                  |
| Weight decay              | 0.1                   |                       |
| AdamW $\beta_2$           | 0.98                  |                       |
| AdamW $\epsilon$         | 1e-6                  |                       |
| Warmup steps              | 3750                  |                       |
| Training steps            | 750k (+ 75k for res. adaptation) |                       |
| Training epochs           | 10 (+ 1 for res. adaptation) |                       |
| $\sigma$ Maximum scale    | 100.0                 |                       |
| Pretraining compute       | TPU v3-512 for 16 days | TPU v3-512 for 5 days |

Table 7: Architecture details, and pretraining hyperparameters, for both model sizes.
the symptoms of training instability – though sometimes the training loss would spike to roughly around the same loss as random initialization, it always converged back to slightly better than it was before the spike. We are not currently sure why this happens.

B.5. Speed improvements during pretraining

We made several high-level algorithmic and engineering implementations to our implementation, which made pretraining run faster, and that we discuss here.

Duplicated video copies. As mentioned in the main text, we create two copies per each video – allowing us to learn separately how to handle audio as an input as well as how to learn from audio. We chose this in part because copying a video does not increase the total compute required by a factor of two. Instead:

1. We encode the underlying video frames and audio clips only once (for the two video copies), and then duplicate the encodings; this is far more efficient than encoding them both separately from scratch.

2. For the two video copies, we sampled two disjoint sets of masks (for which audio and text subsegments are replaced with MASK) at a 25% rate. This increases the pool of negative samples for contrastive learning, again increasing training efficiency.

Reducing memory usage. The memory usage of our Transformer implementation scales quadratically with sequence length, which could pose a problem since we operate on sequences of videos. We split the video into two groups of 8 segments, and encode these each group separately by the joint encoder.

Vectorization. We vectorize all joint transformer inputs together into a single call. During this vectorization, we also encode the transcript (for the transcript-frame matching objective).

We note that this vectorization is incompatible with the Mask LM variant proposed by MERLOT [126]. In this variant, which the authors called ‘attention masking,’ two transformer calls must happen sequentially – first, a language only encoder must encode the inputs and mark down (what is presumably) visually-grounded tokens; second, these tokens are masked for the joint encoder. We found that such an objective was unnecessary when pretraining under our contrastive learning approach, which in turn enabled more efficient pretraining.

We discuss the exact pretraining data formatting technique that we used in the next section.

C. Pretraining Data Formatting: alignment and masking

In this section, we discuss how we turn a video \( \mathcal{V} \) into a (masked) list of segments \( \{ s_t \} \) for pretraining.

Recall that each segment contains a video frame \( v_t \), ASR tokens \( \omega_t \), and audio \( a_t \). We generate the list of segments by iterating through the video with a 5-second sliding window.

Audio and text subsegments for masking. We want audio to be used in part as a target for contrastive prediction. However, during early exploration we found that 5 seconds of audio could correspond to many BPE tokens; roughly 15 on average. We use past work in language modeling as a guide [63, 92] and wanted an average span length of around 5 tokens. To get this, we split each audio segment into three equal subsegments, each with a duration of 1.66 seconds. We can then perform masked language modeling at the aligned subsegment level, where we mask out the text corresponding to an audio subsegment, and have the model (contrastively) predict the masked-out span of text, as well as the corresponding span of audio. We use a masking rate of 25%, which means that a quarter of the subsegments will be corrupted and replaced by a MASK token.

In theory, splitting the videos into (masked) segments ought to be straightforward. However, the key challenge that we ran into is that the YouTube caption timing information is unreliable. Problems might arise when we perform pretraining with both audio and text, on misaligned data. Suppose the model is given audio in segment \( s_{t-1} \) that ends with somebody saying the word ‘pasta.’ If the alignment between audio and text is off, the model might be able to cheat the desired task by simply predicting the word ‘pasta’ for segment \( s_t \) – thereby turning the challenging masked-prediction task into an easier speech recognition task; we discuss this in more detail in Appendix C.1.

### Table 8: Hyperparameters for finetuning on downstream tasks.

<table>
<thead>
<tr>
<th></th>
<th>Base</th>
<th>Large</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>VCR</strong></td>
<td>32</td>
<td>5</td>
</tr>
<tr>
<td>Batch Size</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Training Epochs</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Image Size</td>
<td>288×512</td>
<td></td>
</tr>
<tr>
<td>Learning Rates Tried</td>
<td>1e-5, 2e-5, 3e-5, 8e-6, 1e-5, 1.2e-5</td>
<td></td>
</tr>
<tr>
<td>Learning Rate</td>
<td>2e-5, 8e-6</td>
<td></td>
</tr>
<tr>
<td><strong>TVOA</strong></td>
<td>32</td>
<td>5</td>
</tr>
<tr>
<td>Batch Size</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Training Epochs</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Image Size</td>
<td>288×512</td>
<td></td>
</tr>
<tr>
<td>Learning Rates Tried</td>
<td>5e-6, 1e-5</td>
<td>5e-6, 1e-5</td>
</tr>
<tr>
<td>Learning Rate</td>
<td>5e-6</td>
<td></td>
</tr>
</tbody>
</table>

Sometimes there are long ‘pauses’ in videos where nothing gets said. When this happens – if two segments in a row have fewer than 8 BPE tokens – we merge them 90% of the time, in effect ‘fast-forwarding’ the audio and still extracting a frame from the middle. We do this at most twice, so the total length is at most 15 seconds here (in effect, a ‘playback’ rate of 1x, 2x, or 3x). In roughly 90% of cases, the segments are 5 seconds of length.
One way of addressing the timing issue would be to run our own ASR model over all videos, but we chose not to do this due to computational expense. Instead, we adopted two complementary strategies. First, we trained a lightweight regressor to refine the timing information (C.2); second, we mask audio and text conservatively, to minimize alignment errors (C.3). Finally, we discuss how we combine everything efficiently (in a vectorized way) in C.4.

C.1. YouTube Caption Timings

YouTube provides automatically generated captions for accessibility purposes, which include timing information on each word. In the subtitle encoding that we used (vtt), each word $w$ contains a single timestamp $t$ which corresponds to when the word should flash on-screen. The timings are mostly accurate, but we found two key issues:

a. First, they show up on average roughly 0.1 seconds before each word is spoken, which we suspect might be for usability purposes (perhaps so that while the viewer is reading the caption, they hear the word).

b. Second, with a single timestamp $t$ for each word, it is difficult to infer about pauses. For each word $w$, we can use its timestamp $t$, and the timestamps of adjacent words, to loosely infer an interval $[t_s, t_e]$ around when the word is said. However, the interval is not tight. We can only infer that the word is being actively spoken for some subinterval $[t_s, t_e]$ such that $t_s < t_s < t_e < t_e$. This can lead to high absolute error (in terms of a difference between timestamps), when pauses occur. For example, suppose a speaker says a word, and then pauses. The interval given by the subtitles, $[t_s, t_e]$, might be rather large (possibly a few seconds), even though the actual word was spoken for a fraction of that time.

C.2. Refining timing information

We trained a simple multilayer perceptron regressor to correct the timing information of YouTube transcripts. For data, we used 2000 videos with transcripts from YT-Temporal-180M, and also used Google Cloud’s (highest quality, paid) ASR service to transcribe them. After aligning the words for these transcripts, this gave us tuples of the YouTube ASR word $w$, its provided interval $[t_s, t_e]$, and the ‘ground truth’ interval $[t_s, t_e]$. Our modeling objective was then to predict the desired offsets with respect to the provided interval: $\delta_s = t_s - t_s$ and $\delta_e = t_e - t_e$. We took a feature based approach.

For each input $(w, t_s, t_e)$, we used as features:

i. the length of $w$ in characters,
ii. the length of $w$ in BPE tokens,
iii. whether $w$ is uppercase or not,
iv. the number of vowels in $w$,
v. the number of punctuation characters in $w$,
vi. the value of $t_s - t_e$.

We provided these features as input to the model, as well as the corresponding features for the next word, and the previous word. We z-normalized all features and used a two-layer multilayer perceptron, with a hidden size of 32 and RELU activations. We used a tanh activation at the end to bound the regression. The final predictions for $\delta_s$ (analogously for $\delta_e$) were then given by the following equation:

$$\delta_s = c \tanh(w \cdot h + b_1) + b_2$$

where $h$ is the hidden state, and with learnable parameters $c$, $w$, $b_1$, and $b_2$. The learned bounds mean that, no matter what the input, the model will never predict an offset of above $c + b_2$ (of which it learned for both parameters $c \approx 0.2$ and $b_2 \approx 0.11$, so the offsets can never be above 0.3 seconds). We trained our lightweight regression model using an L1 loss, and used it to correct the timing on all of the transcripts.

C.3. Handling worst-case scenarios in masking, when alignment isn’t perfect

The regressor that we described reduces the average timing error of a transcript, as a preprocessing step, but it is not perfect. Thankfully, however, we find that most of the remaining alignment errors are single words that are slightly misaligned. For instance, for three words $w_t, w_{t+1}, w_{t+2}$, the audio corresponding to the time interval around $w_t$ might contain sound from $w_{t+1}$ being spoken, but rarely $w_{t+2}$. We suspect this is primarily due to the difficulty inferring pauses: by definition, no other word can be said in a pause, so the errors are local.

We present a high level approach for masking audio and text, that in turn addresses these alignment issues (making it difficult for models to cheat). A diagram is in Figure 7.

Recall that in our framework, we only either go from ‘vision and text → text and audio’ (VT→TA), or, ‘vision, text, and audio → text’ (VTA→T). One of the reasons we did this is to avoid allowing a model to cheat by performing speaker identification (or even ‘microphone identification’), which might be feasible if audio was given to the joint model as input. We can handle the two cases separately:

a. **Vision and text → text and audio** (VT→TA). Here, the text as input (to the joint encoder) might overlap with the audio we are trying to predict. Our solution here is thus to donate nearby tokens from the predicted span, to the input. Let the span that we are trying to predict (and that
The audio has alignment errors versus the aligned ASR text. So...

Figure 7: An overview of our masking strategy for dealing with sequences of video frames, ASR, and audio. We have noisy timing information for each word, so we can align the ASR text with audio spans of 1.6 seconds each, using three sub-segments of audio and text for each video frame. However, there exist alignment errors between the ASR and audio sub-segments – certain words (and sub-words) have phonemes that are are in the wrong segment (like ‘back’ in \( w_{1,1} \) is only partially said in the first sub-segment; the ‘k’ sound is said in the second. When audio is only a target, we address these by ‘donating’ tokens to predicted spans. When audio is only provided as input, we address this by sandwiching ‘mask’ tokens between text input (so alignment does not ‘bleed’ over).

We will ‘mask out’) have a start time of \( t_s \) and an ending time of \( t_e \). If the final token in the previous text span, if any, has a timestamp of greater than \( t_s - 0.125 \), we move it to the predicted span; likewise, if the first token in the next text span has a timestamp of less than \( t_e + 0.125 \), we move it to the predicted span as well.

b. Vision, text, and audio → text (VTA→T). In this prediction task, models are given information from all modalities as input, and must predict masked-out text spans. Note that models are only given a single ‘speech’ modality – either text, or audio – at each timestep. What this means is that we can carefully choose which input subsegments to turn into ‘audio subsegments,’ and which to turn into ‘text subsegments.’ Our strategy is, given a masked out subsegment, to turn 80% of adjacent subsegments into ‘text subsegments.’

We give an illustration of this in Figure 7, part 2. Here the word ‘focus’ is part of \( a_{4,1} \) but also \( w_{3,3} \). This might make \( w_{3,3} \) overly easy to predict, if we gave the model \( a_{4,1} \) as input. Our solution is thus to give the model text from \( w_{3,2} \) and from \( w_{4,1} \) as input; we are guaranteed that there is no misalignment overlap here between input and prediction spans. All of the other subsegments (not adjacent to one of the 25% that we mask out) will be provided as audio.

C.4. Putting it all together, along with web text

Finally, we discuss how we combine the various masking approaches into the prediction tasks outlined in the main text. Each video has \( N = 16 \) video segments, and three sub-segments of audio or text spans per segment. We consider two sub-problems for this video sequence:

i. in VT→TA, vision and text are provided as input, and the model must predict masked-out text and audio. These are done on top of separately-encoded \( \text{MASK} \) tokens and \( \text{MASKAUDIO} \) tokens, to enable the model to learn different
predictions for each modality over two separate transformer ‘columns.’

ii. In VTA→T, vision, text and audio are provided as input, and models must predict mask-out text. Here, we use the term ‘predict’ as a shorthand for our contrastive objective – in which a model must match a context (a jointly-encoded \text{MASK}) to the exact missing span in question, where many negative contexts and spans are provided.

We use a masking rate of 25% for audio and text subsegments, and there are 3 subsegments per segment. This means that a single video instance gives us \(48 \times 0.25 = 12\) masked-out spans of text, for each of VT→TA and VTA→T, so 24 in total (as we use disjoint masked-out subsegments). Likewise, it gives us 12 masked-out spans of audio. If we scaled these to the whole batch of 1024 videos, we would have 12k audio span options and 24k text span options. This might suffice, but scaling up the pool of candidates boosts performance in a contrastive setting, as suggested from prior work (e.g. [91]), and as our ablations (Table 1) support as well. Thus, we do the following:

a. Text candidates. We scale up the text candidates by simultaneously training the model on web text, from The Pile [40]. The joint encoder – which can handle pooled video, pooled audio, and BPE-encoded text – is simultaneously given a sequence of web text, for each video that we have. By performing the span-contrastive objective with this piece of web text as well, we can not only teach the model about written (as opposed to spoken) language, but we can scale up the set of text candidates as well.

Let each web-text sequence be of length \(L\). We first divide it into fake regions that ‘look like’ the text subsegments in length. We do this by calculating the empirical length distribution of the text subsegments, and then using this (categorical) distribution to sample a sequence of sub-segment lengths \(\ell_1, \ldots, \ell_K\).\(^7\) We clip the sampled sequence, such that \(\sum_i \ell_i = L\).

Next, we mask the fake subsegments. During pretraining, we use text sequences of length \(L = 800\), but a model sequence length of only 640. Because we are masking spans and not individual tokens, the text sequences ‘shrink’ when we mask them. We extract exactly 38 masked-out spans, which corresponds to around 25\% of total text.

Finally, we combine the target spans that we took from the webtext sequence, with the target spans from the video. We note that sometimes – especially in a video – text spans might be empty. Not every 1.6 second slice of a video has someone speaking. We thus try to not use these empty spans in our contrastive objective. For each video (which is paired with text for implementation reasons) we select the ‘best’ 48 text spans out of the (38+24) options – penalizing empty spans, and choosing spans from videos 4x as often.

These ‘best 48’ text spans, as well as the pooled contexts that they were paired with, will be used in the contrastive objective. Aggregating over the entire batch of 1024 videos (and 1024 web text sequences), this gives us 49152 text spans as candidates, for the all-pairs symmetric softmax between text spans and contexts.

b. Audio candidates. For each video, we note that we have exactly 12 pooled \text{MASK} Audio tokens, where the model is trying to predict the corresponding audio span. One option would be to just use those 12 corresponding audio spans as the targets, aggregate these over the batch, and do a symmetric-cross-entropy loss.

However, we can do even better \textit{for free}. Note that for the VT→T direction, we might have to encode many of the audio spans \textit{anyways}, using the lower level audio encoder (which simultaneously extracts a CLS representation and a sequence-level pooled representation). To simplify implementation, we encode \textit{all 48 audio spans} per video. We can use these audio spans as candidates.

Thus, we do the following when computing the loss over audio prediction. We aggregate all 12288 contexts from the \text{MASK} Audio tokens in the batch, and we aggregate all 49152 candidate audio spans. We perform an all-pairs dot product between these two sets, and use it to compute a symmetric cross-entropy loss over both directions. We did not encounter any trouble using the same temperature for both directions (even though for one direction, there are 12288 options, and for the other, there are 49152).

The combination of these design decisions provide more ‘hard negatives’ for the model during training. We also found that they worked well to reduce wasted computation on a TPU. For each video, the joint transformer uses one \(L = 640\) length sequence for transcript-frame matching, two length-\(L\) sequences for the VT→TA direction (as we break it up into two groups of 8 frames each), two length \(L\) sequences for the VTA→T direction, and finally one length-\(L\) sequence of text. These sequences can all be vectorized together, and the total batch size is \(6 \times \) the number of videos. This is helpful because using an even-numbered batch size reduces wasted computation on a TPU.

D. Downstream Task Implementation Details

In this section, we present information for how we adapted \texttt{RESERVE} on downstream tasks.

\(^7\)The empirical distribution for each length, in order from a length of 1 to 15, is [0.03, 0.05, 0.08, 0.11, 0.13, 0.12, 0.10, 0.07, 0.05, 0.03, 0.02, 0.01, 0.006, 0.003].
D.1. Setup for finetuned Tasks

For adapting RESERVE in a finetuned setting, we take the following approach. We always trained the model for either 3 or 5, depending on the dataset size. We always used a batch size of 32. We use a linear warmup of the learning rate over the first half of the first epoch, with a linear decay thereafter to 0. To find the learning rate, we did a small grid search generally centered around 1e-5; details in Table 7.

When finetuning (and pretraining), we did not use any dropout to make implementation simpler. Instead, as a way to apply regularization, we used the same $L_2$ penalty as in pretraining (a weight decay of 0.1), but with respect to the pretrained weights. This idea was used in [118] among other works, and although it often tends to underperform dropout [72], it is simple to implement. We also randomly resized images (or video frames) to between 100% and 110% of the network’s size. We suspect that additional work in applying other regularization techniques (e.g. AutoAugment [25]) could improve our results.

D.1.1 Visual Commonsense Reasoning

We randomly flip images left or right, so long as there is no instance of the word ‘left’ or ‘right’ in the question, answer, or rationale candidates.

We also followed past work in ‘drawing on’ the provided detection tags to the image [126]. These are unambiguous references to entities that are then referred to in the question, answer, and rationale. For example, text might reference a ‘person1’, which corresponds to an image region. When drawing on these detection tags, we do so in a deterministic way – for example, ‘person1’ always gets the same box color. We determine the box color by hashing the object’s ID (in this case, ‘person1’) and using that to determine the hue. The model learns the connection between boxes with different hues, and the names, during finetuning.

D.1.2 TVQA

TVQA provides models with a video, a question, and five answer candidates; we represent this as five distinct sequences for the model to score (one per candidate). The version of TVQA that we used also gives models annotations for the time region in the video that is being referenced. It is not clear that only using this region would provide enough context to be able to understand what is going on – enough to answer correctly. Thus, for each question, we extract 35 seconds of video around the provided time region. We then provided the model with two numbers corresponding to the time region, relative to the cropped time interval. For example, if the provided timestamp annotation is $[t_0, t_1]$, we use the following region:

\[
    t_c = \frac{t_0 + t_1}{2} \tag{4}
\]

\[
    t_s = t_c - 17.5 \tag{5}
\]

\[
    t_e = t_c + 17.5 \tag{6}
\]

The location of $[t_0, t_1]$ in relative coordinates is then:

\[
    t_0^r = \frac{t_0 - t_s}{t_e - t_s} \tag{7}
\]

\[
    t_1^r = \frac{t_1 - t_s}{t_e - t_s} \tag{8}
\]

We provide models with $t_0^r$ and $t_1^r$, multiplied by 100 and casted to an integer. Thus, an example TVQA instance might look like:

1 to 28 What is Janice Holding on to after Chandler sends Joey to his room? Chandler’s tie. MASK[subtitles or audio]

This text input corresponds to the first ‘segment’ of a video; to it we append subtitles (or audio representations) from seven segments from the provided TVQA video (with accompanying frames).

D.2. Setup and prompting for Zero-shot tasks

Here, we discuss how we set up various tasks for RESERVE in a fully zero-shot setting. In addition to evaluating RESERVE, we also evaluate CLIP [91] in the same zero-shot setting. CLIP is not pretrained on videos, and it cannot jointly encode text. For each task, we construct CLIP’s label space by taking our prompt and substituting in each possible answer option. We average together the logits over all frames, and take a softmax, giving us a distribution over the task-specific label space.

D.2.1 Zero-shot Action Anticipation on EPIC-Kitchens

We study the task of action anticipation from the EPIC-Kitchens dataset [26], a large egocentric video dataset with 700 unscripted and untrimmed videos of cooking activities. In action anticipation, a model must predict a future action that occurs $\tau_a$ seconds after a given video clip. The observed segments are of arbitrary length; we follow prior work [26] and set $\tau_a = 1$.

The model tries to choose the correct noun and verb that happens next, given a list of predefined options for each. We report results on each category using the class-mean top-5 recall.

Zero-shot inference approach. We directly evaluate the pretrained RESERVE on action anticipation to verify the knowledge learned during pre-training. All prior work reported on the official leaderboard use supervision from the in-domain training set, which we do not use at all [45, 38].
For each action segment, we sample at most \( N = 8 \) image frames and their associated audio, with fixed time interval \( t = 2.0 \) preceding it and ending \( \tau_a \) seconds before the start of the action. We append a \( \text{MASK} \) token as the sole text input (at the last frame, after audio is optionally included).\(^{18}\) We create short phrases out of all candidate nouns and verbs, and use that as our label space to simultaneously predict them both. We compute the score for each verb and noun independently by averaging their scores, over all labels for which they appear.

**Results.** We show the full zero-shot action anticipation results in Table 9. We also show our results on the test set here for our best performing model (\( \text{RESERVE-L} \), with audio provided). It gets competitive results on verb and noun prediction – with only 1.6% and 3.3% lower compared to the challenge winner method AVT+ [45], which is fully supervised and use additional object-level annotations. On Unseen Kitchen and Tail Classes, our model outperforms AVT+ on noun and verb. Overall, audio significantly improves the results – \( \text{RESERVE-L} \) (+audio) outperforms \( \text{RESERVE-L} \) with an average 3.0%, which suggests that it is useful for this task.

**D.2.2 Zero-shot Situated Reasoning**

Next, we evaluate on situated reasoning (STAR) [119] which requires the model to capture the knowledge from surrounding situations and perform reasoning accordingly. STAR dataset includes four types of questions, including interaction, sequence, prediction, and feasibility. A model is given a video clip, a templated question, and 4 answer choices.

**Zero-shot inference approach.** For each video clip, we sample \( N = 8 \) image frames uniformly from the video, we also optionally include the video’s sound.

To reduce domain shift between YouTube data – where people don’t typically ask visual questions, and where ASR typically does not insert question marks – we convert the question-answer pair into a statement. We did so using the question-answer templates provided by the author, with the answer replaced by a \( \text{MASK} \). For example, “\( Q: \) What did the person do with the bottle? – \( A: \) Put down.” will be converted to “The person \( \text{MASK} \) the bottle.”.

We put the converted statement into the first frame and use the four candidate answers as a unique label space (that differs from example to example). Like with EPIC-Kitchens, we also evaluate how much audio can help by masking the audio inputs.

**Results.** We show our zero-shot STAR results in Table 4 in the main text. Our base model outperforms all supervised prior work by 3.7%. The model with audio performs better, with average 1.1% improvement. Interestingly, \( \text{RESERVE-L} \) is worse than \( \text{RESERVE-B} \), we suspect the reason is \( \text{RESERVE-L} \) is sensitive to grammar details. Given the previous example, we note that while ‘Put down’ is a valid answer that might make sense both semantically and syntactically, a different answer ‘pick up’ might be flagged by some English speakers as being ungrammatical: the instantiated template would then be ‘the person pick up the bottle.’ We noticed instances of the larger model paying greater attention to these syntax-level details, even though they were not the focus of the task. It does suggest, however, that additional prompting (or label space augmentation) could resolve these issues and increase performance even further.

**D.2.3 Zero-shot LSMDC**

We evaluate our model on Movie Fill-in-the-Blank [95, 81] task, which based on descriptive audio description for the visually impaired. Given a movie clip and an aligned description with a blank in it, the task is to fill in the blank with the correct word. Following [81], we report prediction accuracy in test set of 30,354 examples from 10K movie clips.

**Zero-shot Inference approach.** We sample \( N = 8 \) video segments uniformly over the movie clip, and extract the audio and middle frame of each segment. We replace the ‘blank’ token in each description with a \( \text{MASK} \) token, and provide it (as text-based input) to the model at its final segment. For the other segments, we optionally provide the model with audio; for all segments, we provide the associated image frame. We use the vocabulary set in the LSMDC dataset as our label space (for what the ‘missing word’ might be).

**Results.** Our results are shown in Table 4 in the main text. Our model obtains 31% when audio is included, which

![Table 9: RESERVE gets competitive results on EPIC Kitchen Action Anticipation challenge with zero-shot, over methods from prior work.](image-url)
outperforms human text-only performance (30.2%) [81],
predicted by human annotators. A supervised LSTM obtains
34.4% in this text-only setting [81] which suggests that there
is a certain textual bias in this task, which our model cannot
learn (as it is zero-shot). This also suggests that state-of-the-art
supervised models exploit patterns in this vocabulary
distribution.

Without such an advantage, our model performs well,
outperforming CLIP (2%) by a large margin. This suggests
that jointly reasoning over both the visual situation, and the
linguistic context of the provided sentence, is helpful for
zero-shot performance on LSMDC fill-in-the-blank.

**D.2.4 Zero-shot MSRVTTQA**

Finally, we evaluate our model on MSR VTT-QA, a question-
answering task over videos [120]. We provide a model with
\(N = 8\) video segments sampled uniformly from the video
clip, and extract an image from each one. For the first seven
segments, we optionally include audio extracted from that
point; at the last segment, we insert a converted version of
the question, along with a \_ MASK. We compare the similarity
of that hidden state to the top 2000 most common answers,
similar to past work [126].

Similar to STAR, we convert the questions into statements
to minimize drift away from the pretraining distribution. We
use GPT3 prompted with several examples for this. Our
exact prompt is the following:

```
Input: what is a car being driven through?
Output: a car is being driven through _.
```

```
Input: who are running across screen?
Output: _ are running across screen.
```

```
Input: when is a girl performing?
Output: a girl is performing at _.
```

```
Input: what is a cartoon doing?
Output: a cartoon is _.
```

```
Input: how many women talk in a bedroom?
Output: _ women talk in a bedroom.
```

```
Input: what a man playing while dancing with others?
Output: a man is playing _ while dancing with others.
```

```
Input: where is a flag hoisted?
Output: a flag is hoisted in _.
```

```
Input: who talks to another man on the couch?
Output: _ talks to another man on the couch.
```

```
Input: what does a teenage girl try to get at a public restroom?
Output: a teenage girl tries to get _ at a public restroom.
```

```
Input: when do the models walk as the audience watches?
Output: the models walk as the audience watches at _.
```

```
Input: what shows a person killing animals in a green forest?
Output: _ shows a person killing animals in a green forest.
```

```
Input: who does a man ask to go on a date?
Output: a man asks _ to go on a date.
```

```
Input: what are three people sitting on?
Output: three people are sitting on _.
```

Then, given a new question \{question\}, GPT3 generates a converted output, wherein we can replace it’s
underscore with a \_ MASK. GPT3 works well at this conversion,
though sometimes it generates a sentence where inserting the
‘correct answer’ feels grammatically strange. For example, the
question ‘how many women talk in a bedroom?’ suggests
any integer might be a reasonable answer. On the other hand,
‘women talk in a bedroom’ implies that ‘one’ is not a valid
answer (since ‘women’ is plural). We note that the errors
caused by this conversion technique are specific to English
grammar, and so if such a question-conversion approach was
done in other languages, there could be more (or less) errors
that directly result.

Our results are shown in Table 4. Of note, our model
through automatic question-conversion outperforms Just Ask
[122], which performs an analogous (supervised-guided)
question conversion on all its YouTube transcripts, before
pretraining. Our model also outperforms CLIP, which cannot
naturally handle dynamic situations.

## E. Dataset Collection

In this section, we discussed how we curated data for
YT-Temporal-1B. We had several goals in mind. We wanted
to use only public-facing data, which motivated our choice
of YouTube as it is a public platform that users understand
is public [64]. We wanted to use this platform to examine to
what extent we can learn multimodal neural script knowledge
from web data alone.

Our data collection strategy in this work was informed
by past work, notably MERLOT [126]. That paper found
that increasing the diversity and scale of a video corpus both
allowed for better learned representations. At the same time,
the data collected by MERLOT (YT-Temporal-180M) has
issues. Of note, the authors’ scraping strategies – to prioritize
monetized content – also led to a lot of U.S. local news being
in that corpus (roughly 30% of all data). Local news might
be problematic to learn from, particularly in that quantity,
due to its numerous biases (e.g. racist coverage on ‘crime’
[44, 31, 30, 57]). Our goal was to expand the dataset in both
diversity and size to 20 million videos, while having less
local news and without scraping private content.

### High level approach.

We adopt a similar dataset collection
strategy as in MERLOT [126]. In the first phase, we
identify a candidate set of videos ID to download. In the
second phase, we open each video ID in YouTube and apply
several filtering steps that go from inexpensive to expensive.
The filtering steps allow us to exit early and possibly avoid
downloading the video if the video seems unsuitable for our
purpose from the title, description, and captions alone.
E.1. Candidate video IDs

For MERLOT’s YT-Temporal-180M, the bulk of the video IDs were identified by applying breadth-first-search on YouTube channels from HowTo100M [84] and VLOG [36]. Each channel often links to other channels, and given a channel it is inexpensive to obtain a list of all its videos using the youtube-dl Python package.

In this paper, we considered numerous approaches to search for diverse, visually grounded videos. We ended up using an approach where we used YouTube’s recommended videos algorithm to suggest similar videos to YT-Temporal-180M. We went through all non-news and non-sports videos YT-Temporal-180M, and opened each video up in YouTube. For each other video that YouTube recommended, we retrieved its channel ID – giving us access to not just that video, but all other videos. This approach yielded 2 million channels, with 200 million videos among them.

E.2. Filtering video IDs by channel

Given this (large) list of channels, each with many videos, we took steps to filter it further. We used the python cld3 library to remove channels whose titles might not be in English. We then finetuned, and used, a language model to identify channels likely to have visually grounded videos, which we describe next.

In more detail, we selected 2000 videos, and asked workers on Mechanical Turk to rate their level of groundedness, their genre, and whether they had explicit content or not. The questions we asked are shown in Figure 8. We annotated 2k videos under this schema, and trained a model to predict the annotations given video metadata.

For model training, we used a slightly different setting to what we gave the crowdworkers. We trained a model to predict the labels, given a formatted list of 5 video titles from the same channel. During training, we made the weak-supervision assumption that all videos from a channel have exactly the same rating (as the video we annotated). This enabled us to collect 84k examples from our 2k annotations. The model we chose was T5-base model [92], which generates the labels left-to-right in text form (and which we converted automatically to a structured representation).

We then used this model to identify channels that seem especially promising. For each channel with at least 5 videos, we randomly sampled 8 sets of length-5 videos, and used the finetuned T5 model to classify them. We filtered out any channel that had at least 25% of likely non-English or irrelevant-English videos, any channel that had at least 25% of slideshows, and any channel that likely had racist or sexist content.

One side benefit of this model is that it allowed us to estimate our videos’ genre breakdown before downloading them. We found 11% Gaming videos, 11% News videos, 20% How-To videos, 20% ‘chatting’ videos, 5% sports videos, 5%

E.3. Filtering videos from their metadata

Last, we filtered and downloaded these videos using a filtering approach similar to [126]. We first retrieved the
video metadata and used it to filter out ‘gaming’ videos. We then retrieved the video’s transcript, and filtered out any video without a ‘dense’ span of spoken words – defined as an interval of 30 seconds where at least 50 words are spoken. Additionally, we used the Python package cld3 to filter out any transcript with a probability of less than 80% of being English. Last, we used a hidden feature in the YouTube API to download thumbnails of the video. Using the image classification model from [126], we filtered out videos whose four thumbnails had an average cosine similarity of above 85%, or that contained fewer than 1 object from COCO.

Unlike [126], we did not use a sequence-to-sequence model to ‘translate’ spoken text to text that appears more stylistically like written English (i.e., by adding capitalization and punctuation, and removing filler words).

F. Additional Experiments and Exploration

In this section, we briefly include additional experiments, showcasing our model’s performance on specific tasks that do not necessarily require multimodal script knowledge.

F.1. Zero-shot Audio classification

We evaluate RESERVE on the task of zero-shot audio classification, to study to what extent its learned audio representations can directly predict text-based labels. We conduct this evaluation on environmental sounds from ESC50 [89], urban sounds from US8K [97], and (as part of the privacy-minded exploration in Appendix A) celebrity voices from VoxCeleb2 [86].

We consider the format where we encode an audio input into a CLS level representation, and retrieve the most-similar label given a set of encoded options. We encode the audio input with our encoder, which takes in as input audio clips of length at most 1.6 seconds. For shorter audio clips (like many sounds in ESC50), we repeat them in time until their length is at least 1.6 seconds. For longer audio clips, we encode multiple CLS representations and then average the resulting vectors.

We consider the following ways to encode the labels:

a. Text-only. Inspired by the prompt ‘a photo of’, which is used in CLIP’s zero-shot image classification task [91], we give RESERVE’s joint encoder a blank image, with associated tokens the sound of $\{$Label$\}. We do this once for each label, giving us a single ‘target’ vector for each possible label in the dataset.

b. Image-only. Inspired by YouTube videos of sound effects, we created image-only prompts that suggest a sound (of the target class) is playing in the background. An example is shown in Figure 9. We encode each image with our joint encoder, and do this once for each label.

We note that for VoxCeleb2, we use face images of celebrities rather than this image-based prompt, due to our interest in exploring whether models can perform person-level recognition due to the privacy issue (Appendix A.1.1).

c. Image and text. Here, we combine both of the above options: encoding one input for each label, using both the image and text prompt.

For each prompt, we append the token ‘\texttt{MASKAUDIO}’ and extract the hidden state from there, as our final representation for that label.

We present our results in Table 10. The results show, possibly surprisingly, that RESERVE can perform optical character recognition over image prompts like Figure 9 – given just the image, its accuracy on ESC50 is higher than given just text. Its accuracy on ESC50 and US8K improves further when given both an image and text.

These results are slightly different for VoxCeleb2, which emphasizes long-tail recognition of people – something that might be more encyclopedic than semantic, and that we did not wish to optimize in this work. There, when given an image of a celebrity’s face, it demonstrates some capacity.

Table 10: Zero-shot audio classification accuracies (%) on ESC50 [89], US8K [97], and VoxCeleb2 [86]. We compare our model with AudioClip [53], which was pretrained on supervised data from AudioSet [42]. Our RESERVE performs well across the board, especially when given both the image and the text as a prompt – demonstrating its OCR capability.

<table>
<thead>
<tr>
<th>Model</th>
<th>Prompting</th>
<th>ESC50</th>
<th>US8K</th>
<th>VoxCeleb2</th>
</tr>
</thead>
<tbody>
<tr>
<td>AudioClip</td>
<td>Text-only</td>
<td>41.6</td>
<td>60.2</td>
<td>10.8</td>
</tr>
<tr>
<td></td>
<td>Image-only</td>
<td>42.8</td>
<td>54.3</td>
<td>13.3</td>
</tr>
<tr>
<td>RESERVE-L</td>
<td>Image and text</td>
<td>52.2</td>
<td>62.3</td>
<td>9.6</td>
</tr>
</tbody>
</table>

---

99For instance, youtube.be/VimgKryu4__k.
we'll be seasoning it with this homemade adobo seasoning. I couldn't find it, so I'm gonna take these two chicken breasts and season them with the adobo. I just have a little bit of olive oil heating up in a skillet, and some garlic bread.

I served it with a little side salad of... Figure QPZ MASK... shells recipe and boy have I been missing out because... that's still um... condition is when I bought it... we had... Queensland... people just keep coming back year after year, but now it's getting extremely hard with the... affordable weekend and people come back... three basically for the parental... 

Hey y'all welcome back to another week of what's for dinner. It is Monday and I made back I think basically for the parental... innocent people is here......

... people just keep coming back year after year, but now it's getting extremely hard with the... we had... Queensland... that's still in original condition is when I bought it... and a repaint there at one stage. Other than just some running maintenance, we've created to pay their paper and have the student go... we'll have a paper we use the same rubric little writing rubrics for our writing units. We create our... corrections anything you'd like to share well...

... differently in using rubrics developing rub... corrections anything you'd like to show well... little writing rubrics for our writing units. We create our... corrections anything you'd like to share well... little writing rubrics for our writing units.

I'll have a appear we use the same rubric... we've created to pay their paper and have the student go...

... they all welcome back to another week of what's for dinner. It is Monday and I made... shell trick year actually my first time ever making a stuffed... shell recipe and boy have I been missing out because... served it with a little side salad of... and some garlic... it... the pins bend... Now I just have a little bit of olive oil heating up in a skillet, and then I'm gonna take these two chicken breasts and I'm gonna [MASK]... I can't be seasoning it with this homemade adobo seasoning I couldn't find it.