# POLYVIT: CO-TRAINING VISION TRANSFORMERS ON IMAGES, VIDEOS AND AUDIO

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Can we train a single transformer model capable of processing multiple modalities and datasets, whilst sharing the majority of its learnable parameters? We present PolyViT, a model trained on image, audio and video which answers this question. By co-training different tasks on a single modality we are able to improve the accuracy of each individual task and achieve state-of-the-art results on 5 standard video- and audio-classification datasets. Co-training PolyViT on multiple modalities and tasks leads to a model that is even more parameter-efficient, and learns representations that generalize across multiple domains. Finally, we show that co-training is simple and practical to implement, as we do not need to tune hyperparameters for each combination of datasets, but can simply adapt those from standard, single-task training.

## 1 INTRODUCTION

Transformers (Vaswani et al., 2017) are a flexible architecture which operate on a sequence of input tokens. While it was originally designed for natural language processing, it has recently been adapted to a range of perception tasks, such as classification of images (Dosovitskiy et al., 2021), video (Arnab et al., 2021) and audio (Gong et al., 2021; Nagrani et al., 2021). Despite recent advances across different domains and tasks, current state-of-the-art methods train a separate model with different model parameters for each task at hand.

In this work we show how to train a single, unified model (Fig. 1) that achieves competitive, or state-of-the-art results for image-, video- and audio-classification. We go beyond using a common architecture for different modalities (Jaegle et al., 2021), as we also share model parameters across tasks and modalities, thus enabling potential synergies.

Our main technique is *co-training*: training a single model on multiple classification tasks (across potentially multiple modalities) simultaneously. We consider various settings, and simultaneously solve as many as 9 different image-, video- and audio-classification tasks. As shown in Fig. 1, our model is capable of performing multiple tasks, but performs a single task at a time for a given input. Although similar techniques have been explored in computer vision (Maninis et al., 2019) and natural language (Raffel et al., 2019), we are not aware of previous work that have considered multiple modalities and achieved state-of-the-art results with this approach.

We show that our co-training setup has multiple benefits: In particular, it is parameter-efficient as we share the transformer parameters for each of the $n$ tasks of interest, approximately reducing the number of parameters by a factor of $n$. This has practical advantages when deploying models on edge devices with limited memory. Furthermore, co-training on tasks of the same modality leads to accuracy improvements on each individual task whilst also linearly decreasing total parameters. In particular, we achieve state-of-the-art results on video and audio classification across 5 different datasets. In addition, when we extend co-training to multiple tasks and modalities, we observe that our accuracy is still competitive with the state-of-the-art whilst being even more parameter-efficient – our model trained on 9 datasets uses 8.3 times fewer parameters whilst having at most a 1.2% accuracy drop compared to state-of-the-art single-task baselines. Finally, linear probing experiments show that this multi-task, multi-modal model is able to learn representations that generalize across multiple tasks and domains.

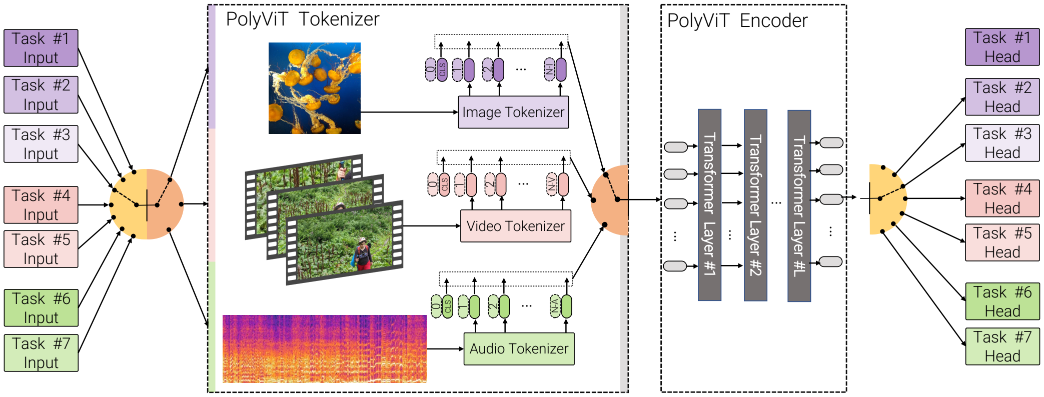

Figure 1: Overview of PolyViT. Our model is capable of performing multiple tasks spanning different modalities, and processes a single task at a time. The architecture consists of a transformer encoder shared among all tasks, modality-specific input tokenizers and task-specific output heads.

In addition to all the benefits outlined above, our co-training setup is simple and practical to implement. It does not require hyperparameter tuning for each combination of datasets, as we can readily adapt the settings of standard, single-task training. In addition, co-training does not increase the overall training cost either, as the total number of training steps does not exceed that of the sum of each single-task baseline. For reproducibility, we will release code and checkpoints.

## 2 Preliminaries

We define a *modality* as the type of input processed by the network. In this work, we consider images, audio, and video (specifically, the sequence of image frames in a video) as three separate modalities. We perform classification as it is a fundamental problem whose solutions are often extended to more complex ones (Girshick et al., 2014; He et al., 2017). By *task*, we refer to a pair of input modality and a set of classes from which one or multiple classes are to be selected for a given input. Each task corresponds directly to a dataset, for example, ImageNet-1K (Deng et al., 2009) for image classification or Kinetics 400 (Kay et al., 2017) for video classification.

### 2.1 Vision Transformers and extensions

The Vision Transformer (ViT, Dosovitskiy et al. 2021) is a transformer-based architecture for image classification that closely follows Vaswani et al. (2017). In contrast to language which is intuitively tokenized into words, ViT extracts tokens from the input image, $\mathbf{x}^{\text{IMG}} \in \mathbb{R}^{H \times W \times 3}$, by splitting it into $N = \lfloor H/h \rfloor \times \lfloor W/w \rfloor$ non-overlapping patches, $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathbb{R}^{h \times w \times 3}$. Each patch, $x_i$, is then projected into a token $\mathbf{z}_i \in \mathbb{R}^d$ by a linear operator $\mathbf{E}$, $\mathbf{z}_i = \mathbf{E}\mathbf{x}_i$ (*input embedding operator*). All tokens are then concatenated into a sequence, which is also prepended with a learnable *class token* $\mathbf{z}_{cls} \in \mathbb{R}^d$. Learnable *positional embeddings* $\mathbf{p} \in \mathbb{R}^{N \times (d+1)}$ are also added to this sequence as the transformer is otherwise permutation invariant. We denote this tokenization process as

$$\mathbf{z}^0 = [\mathbf{z}_{cls} \quad \mathbf{E}\mathbf{x}_1 \quad \ldots \quad \mathbf{E}\mathbf{x}_N] + \mathbf{p}. \tag{1}$$

Note that the linear operator $\mathbf{E}$ can also be thought as a 2D convolution with kernel of size $h \times w$ and strides $(h, w)$. The sequence of tokens, $\mathbf{z}$, is then processed by a transformer encoder, consisting of $L$ layers. Each layer, $\ell$, is applied sequentially, and performs the transformations,

$$\mathbf{y}^\ell = \text{MSA}\left(\text{LN}\left(\mathbf{z}^{\ell-1}\right)\right) + \mathbf{z}^{\ell-1} \tag{2}$$

$$\mathbf{z}^\ell = \text{MLP}\left(\text{LN}\left(\mathbf{y}^{\ell}\right)\right) + \mathbf{y}^{\ell}, \tag{3}$$

where MSA is the multi-head self-attention operation (Vaswani et al., 2017), MLP is a neural network with a single hidden layer and a GeLU nonlinearity (Hendrycks & Gimpel, 2016), and LN denotes layer normalization (Ba et al., 2016).

For a $C$-class classification problem, the class probability logits produced by the model are obtained by applying an output *linear head* on the encoded classification token, $\mathbf{z}^L_{cls}$, as

$$\mathbf{W}_{out}\mathbf{z}^L_{cls} + \mathbf{b}_{out} \in \mathbb{R}^C, \tag{4}$$

where $\mathbf{W}_{out} \in \mathbb{R}^{C \times d}$ and $\mathbf{b}_{out} \in \mathbb{R}^{C}$ are the linear head's learnable parameters.

**Extensions of ViT to audio and video** The *Audio Spectrogram Transformer* (AST, Gong et al. 2021) follows the same architecture as ViT, with the only difference that its inputs are log-mel spectrograms. Spectrograms are image-like, time-frequency representations of audio, and can be tokenized in the same manner as images. Moreover, the best AST model was initialized from ViT models pretrained on large image datasets.

*Video Vision Transformers* (ViViT, Arnab et al., 2021) are an extension of ViT to video. The authors proposed four model variants, and we consider the unfactorized version (Model 1 in Arnab et al., 2021). This model differs from ViT only in the input tokenization process, which it extends from 2D image patches to 3D spatio-temporal "tubelets". Namely, a video input $\mathbf{x}^{\text{VID}} \in \mathbb{R}^{F \times H \times W \times 3}$ is split into $N = \lfloor F/f \rfloor \times \lfloor H/h \rfloor \times \lfloor W/w \rfloor$ non-overlapping tubelets $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathbb{R}^{f \times h \times w \times 3}$. Following ViT, a linear operator $\mathbf{E}^{\text{VID}}$, which can be interpreted as a 3D convolution, projects $\{\mathbf{x}_i\}$ into a sequence of tokens $\{z_i = \mathbf{E}^{\text{VID}}\mathbf{x}_i \in \mathbb{R}^d\}$, and computations (1-4) are repeated.

**Initialization** Finally, note that ViT, ViViT and AST all achieve their highest performance when pretrained on a large-scale dataset such as ImageNet-21K (Deng et al., 2009) or JFT (Sun et al., 2017). More specifically, ViT was initially pretrained on ImageNet-21K or JFT, and then finetuned at higher resolution on target datasets such as ImageNet-1K. ViViT and AST also initialize their models from large-scale, image-pretrained models. In all of these cases, the positional embeddings, $\mathbf{p}$, which depend on the sequence length $N$ (and thus the input resolution), are interpolated from the pretrained model to the finetuned model. Furthermore, the 3D embedding projection of ViViT, $\mathbf{E}^{\text{VID}}$, is initialized from the 2D projection of ViT, $\mathbf{E}^{\text{IMG}}$ (Arnab et al., 2021).

The similarities between ViT, ViViT and AST allow us to construct a multi-modal model with a shared transformer encoder, and separate input tokenizers as described next in Sec. 3.

# 3 Co-training ViT on images, audio and video

## 3.1 PolyViT architecture

PolyViT is a single architecture that is capable of processing inputs from multiple modalities. As shown in Fig. 1, we share a transformer encoder among different tasks and modalities, enabling up to a linear reduction in parameters with the number of tasks. Note that PolyViT with $L$ layers acts like an $L$-layer ViT when processing images, an $L$-layer AST when processing audio and an $L$-layer unfactorized ViViT when processing video. And whilst it is capable of handling multiple modalities, it performs one task from one modality in a given forward pass.

As shown in Fig. 1, PolyViT employs modality-specific class tokens, $\mathbf{z}_{cls}^{\text{IMG}}, \mathbf{z}_{cls}^{\text{VID}}, \mathbf{z}_{cls}^{\text{AUD}}$, input embedding operators, $\mathbf{E}^{\text{IMG}}, \mathbf{E}^{\text{VID}}, \mathbf{E}^{\text{AUD}}$, and positional embeddings $\mathbf{p}^{\text{IMG}}, \mathbf{p}^{\text{VID}}, \mathbf{p}^{\text{AUD}}$. This allows the network to encode modality-specific information that can be leveraged by the subsequent, shared transformer backbone. It also accounts for the fact that the number of tokens per modality may vary.

A separate output linear head (Eq. 4) is then used for each task as shown in Fig. 1. Each head has learnable weights,

$$\mathbf{W}_{out} = \mathbf{W}_{out}^{mod,j} \in \mathbb{R}^{C_j \times d}, \quad \mathbf{b}_{out} = \mathbf{b}_{out}^{mod,j} \in \mathbb{R}^{C^j}, \tag{5}$$

where $mod \in \{\text{IMG}, \text{VID}, \text{AUD}\}$ is a modality, $j \in \{1, \ldots, T^{mod}\}$ is a task index in the set of tasks for that modality, $T^{mod}$ is the number of tasks for that modality and $C_j$ is the number of classes for that task. Note that the output heads are the only task-specific parameters. The input embedding operators, positional embeddings and class tokens are shared by all tasks within a modality.

To increase model capacity when co-training on a large number of tasks and modalities simultaneously, we can optionally include $L_{adapt} \geq 0$ modality-specific transformer layers (which we denote as *modality-adaptor layers*). These transformer layers are applied directly after tokenization. In this case, there are $L_{shared} = L - L_{adapt}$ layers which are shared among all modalities and tasks. We can think of this case as using a more shallow transformer encoder, but a deeper subnetwork to extract tokens from different modalities.

As almost all computation and parameters within our architecture are within the $L$ layers of the transformer encoder, if there are $n$ tasks, we reduce the total number of parameters by a factor of approximately $n$ when $L_{shared} = L$. This is in comparison to standard, single-task training. Note that the overall inference time does not change, as PolyViT still performs one task per forward pass.

### 3.2 Co-training procedure

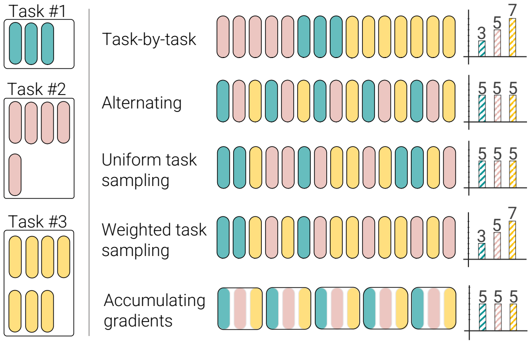

Figure 2: Task sampling schedules considered in this paper. Each element within a task corresponds to the number of training steps performed for that task by the baseline model.

We optimize all PolyViT model parameters, $\theta$, simultaneously across all the tasks that we are co-training on with stochastic gradient descent (SGD). As a result, there are a myriad of design choices on how to construct training batches, compute gradients to update model parameters, and which training hyperparameters to use.

In all cases, we construct our training minibatches using examples from a single task. This design choice allows us to evaluate gradients and perform a parameter update using the same training hyperparameters (e.g., learning rate, batch size, and momentum) as a conventional single-task baseline. As a result, we can perform co-training on multiple tasks without any additional hyperparameter tuning compared to the single-task baseline (Dosovitskiy et al., 2021; Gong et al., 2021; Arnab et al., 2021), making co-training simple to perform in practice, and alleviating the need to perform large hyperparameter sweeps in order to achieve competitive accuracy. Moreover, constructing minibatches from a single task (where each example has the same number of tokens) has computational advantages on GPU- or TPU-accelerators too, as tokens do not need to be padded to a maximum sequence length.

During co-training, for each SGD step, we sample a task (dataset), then sample a minibatch from that task, evaluate a gradient and then perform a parameter update. An important consideration is the order in which we sample tasks and whether we accumulate gradients over different minibatches and tasks. We describe several task sampling schedules below and in Fig. 2. We first denote $U_j$ as the number of SGD steps for the single-task baseline that the original authors reported for their best model, where $j \in \{1, \ldots, T\}$ indexes the task and $T = T^{\text{IMG}} + T^{\text{AUD}} + T^{\text{VID}}$ is the total number of tasks. Furthermore, we define $U$ as the total number of SGD steps during co-training.

**Task-by-task** In this deterministic schedule, the first $U_{j_1}$ SGD steps are performed with task $j_1$, the next $U_{j_2}$ steps using task $j_2$ and so on, where $[j_1, \ldots, j_T]$ is a random task order.

**Alternating** This deterministic schedule alternates between tasks in a fixed, repeating order. Concretely, we perform a single SGD step for each task in sequence before repeating the same order. We set $U = \sum_{j=1}^{M} U_j$ which implies $U/T$ training steps per task.

**Uniform task sampling** This is a stochastic version of the schedule above, where the task for each SGD step is sampled from a uniform distribution, with probability $1/T$. We implement it such that the number of training steps for task $j$ is exactly $U/T$, by randomly permuting an array with $U$ elements, where $U/T$ elements correspond to each task.

**Weighted task sampling** In this schedule, we sample each task with a weight proportional to the number of training steps in the single-task baseline. Therefore, $U = \sum_{j=1}^{M} U_j$, and the sampling weight for task $j$ is $U_j/U$. We implement this schedule, using the same implementation as above, to ensure that we perform exactly $U_j$ steps for task $j$.

**Accumulating gradients** For $T$ tasks, we perform a forward and backward pass on a minibatch for each task, summing the gradients over each task. We then perform a single parameter update with the accumulated gradients, thus effectively using a larger batch size encompassing all the tasks being co-trained. Here, we set $U = (\sum_{j=1}^{T} U_j)/T$.

### 3.3 Initialization of PolyViT

As described in Sec. 2.1, ViT, ViViT and AST models are initialized from models pretrained on ImageNet-21K or JFT before being finetuned for the task of interest. In all of our experiments, we

Table 1: The effect of the task sampling schedule on co-training performance on multiple modalities and tasks. The highest accuracy is shown in bold, and the second-highest is underlined. Note how the "Weighted" task sampling method consistently achieves the highest accuracy for 8 out of 9 tasks, and second-highest on the remainder. Results are on the validation set.

| | Image | | | | | Video | | Audio | |
|---|---|---|---|---|---|---|---|---|---|
| Schedule | Im1K | C100 | C10 | Pets | R45 | K400 | MiT | MiniAS | VGG |
| Task-by-task | 0.3 | 0.8 | 11.7 | 1.9 | 2.0 | 0.3 | 0.3 | 1.6 | 37.2 |
| Accumulated | **88.1** | 90.0 | 98.8 | 94.0 | 96.1 | 58.0 | 22.5 | 22.9 | 27.3 |
| Alternating | 86.0 | 89.4 | 99.2 | 94.0 | 95.8 | 69.7 | 30.0 | 31.4 | 44.6 |
| Uniform | 85.8 | 89.3 | 98.6 | 94.6 | 96.1 | 68.8 | 29.3 | 30.6 | 44.1 |
| Weighted | 86.9 | **90.4** | **99.3** | **96.5** | **97.0** | **71.6** | **32.5** | **33.5** | **49.2** |

also finetune from a ViT model pretrained on ImageNet-21K unless otherwise stated, and follow the initialization methods for the positional embeddings, $\mathbf{p}$, and input embeddings, $\mathbf{E}$, for each modality as described in Dosovitskiy et al. (2021) and Arnab et al. (2021).

When we use modality-adaptor layers, that is $L_{adapt} > 0$, the first $L_{adapt}$ layers for each modality are initialized with the same first $L_{adapt}$ layers from the pretrained ViT model. These parameters are however allowed to change from each other during training. Similarly, shared PolyViT layers are initialized from the last $L_{shared}$ transformer encoder layers from the pretrained ViT model. Note that the output linear heads are all initialized randomly.

## 4 Experiments

### 4.1 Experimental Setup

We train PolyViT simultaneously on 9 diverse classification tasks spanning the image, video and audio modalities. Note that datasets and tasks have a one-to-one correspondence. We chose this setup of 9 tasks, as the datasets include a large variation in domain and training set sizes. Furthermore, the single-task baseline training hyperparameters vary substantially between the tasks. Consequently, we believe this presents a challenging co-training setup.

When co-training for image classification, we use ImageNet-1K, CIFAR-10 and -100, Oxford-IIIT Pets and RESISC45. For video, we use Kinetics 400 and Moments in Time, and for audio, AudioSet and VGGSound. Exhaustive details of these datasets are in Appendix A. As in Nagrani et al. (2021), we evaluate on the whole AudioSet validation set, but use a smaller balanced subset referred to Mini-AudioSet (MiniAS) for initial experiments. We then use a larger, balanced subset of 500 000 examples (referred to AS-500k) for our state-of-the-art comparisons following Nagrani et al. (2021). We follow standard evaluation protocols for each task, reporting classification accuracy (%) for all tasks except AudioSet, where we report mean average precision (mAP) as it is a multilabel problem.

We set the training hyperparameters for these tasks (and those of the single-task baselines) using the values reported by Dosovitskiy et al. (2021) for image tasks, Arnab et al. (2021) for video tasks and Nagrani et al. (2021) for audio tasks (detailed in Appendix A). Note that the "audio-only" model of Nagrani et al. (2021), which we use as our baseline, is identical to AST (Gong et al., 2021), and we choose it since the authors have evaluated on more datasets.

We perform experiments with two standard transformer encoder configurations: Base (number of layers, $L = 12$, hidden dimension $d = 768$, attention heads $h = 12$) and Large ($L = 24$, $d = 1024$, $h = 16$) following (Devlin et al., 2019; Dosovitskiy et al., 2021). As in Dosovitskiy et al. (2021), we initialize our PolyViT model and baselines with ViT pretrained on ImageNet-21K. We refer to this initialized model as ViT-Im21K. For reproducibility, we will release code and models, and include exhaustive experimental details in Appendix A.

### 4.2 Selecting the best task sampling schedule for co-training

We begin by analyzing the effect of the different task sampling schedules listed in Sec. 3.2. We use the full, aforementioned 9-task set-up with PolyViT-Base and all encoder layers shared ($L_{shared} = L = 12$, $L_{adapt} = 0$).

Table 2: Co-training with PolyViT-Base. As indicated by the "#Models" column, some rows correspond to multiple trained models. In this case, we report the total number of parameters across all the models. PolyViT co-trained on a single-modality outperforms single-task baselines in most cases, whereas PolyViT co-trained on multiple modalities achieves competitive performance with a large reduction in parameters. Results are on the test set. Further dataset details in Appendix A.

| | | | Image | | | | | Video | | Audio | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Model | #Models | #Params | Im1K | C100 | C10 | Pets | R45 | K400 | MiT | MiniAS | VGG |
| ViT-Im21K Linear probe | 1 | **93M** | 80.7 | 76.2 | 91.7 | 91.8 | 81.7 | 64.0 | 25.5 | 11.3 | 15.7 |
| Single-task baseline | 9 | 773M | 83.1 | 92.0 | 99.0 | 94.5 | **96.7** | 78.7 | 33.8 | 29.3 | **51.7** |
| PolyViT, 1 modality | 3 | 263M | **84.3** | **93.3** | **99.1** | **95.1** | 96.4 | **80.2** | **36.5** | **36.7** | 51.6 |
| PolyViT, $L_{adapt} = 0$ | 1 | **93M** | 83.1 | 91.2 | 99.0 | 95.0 | **96.7** | 77.5 | 33.2 | 32.3 | 50.6 |
| PolyViT, $L_{adapt} = L/2$ | 1 | 178M | 82.8 | 91.5 | 99.0 | 95.0 | 96.6 | 79.4 | 35.3 | 33.1 | 51.5 |

As shown in Tab. 1, the "Task-by-task" schedule performs poorly, and only achieves decent performance on one task, as it suffers from catastrophic forgetting (French, 1999). The "Accumulated" sampling strategy requires using a single learning rate for all tasks (since the accumulated gradient over all tasks is used for performing a parameter update). As we used a learning rate of 0.03, which is the learning rate used by the image tasks, and significantly lower than the learning rates for the video and audio tasks of the baselines (details in Appendix A), this method only performs well on the image datasets. The "Alternating", "Uniform" and "Weighted" strategies perform the best, showing that task-specific learning rates, and switching between gradient-updates for different tasks is crucial for co-training performance.

In particular, the "Weighted" sampling method performs the best, achieving the highest accuracies on 8 of the 9 tasks (and second-highest on the remainder), motivating us to use it for all subsequent experiments. Note that the "Weighted" strategy samples tasks with a lower number of training steps in their baseline training configurations less frequently. In particular, the Pets task is only sampled for 500 iterations, out of the 417 000 total steps, or just 0.11% of the SGD updates. Nevertheless, it still achieves the highest accuracy on this task. Another advantage of the "Weighted" strategy is that it performs the same number of steps per task as a single-task baseline. Therefore, it uses the same computational resources during training as 9 separate, single-task baselines. Our experiment also shows that if we do not have training hyperparameters for a new task, we can simply tune them separately in the single-task setting, and then reuse them for co-training. This approach requires significantly less computation than tuning training hyperparameters directly in the co-training setup.

### 4.3 Co-training with PolyViT

Table 2 presents approaches for training models to solve 9 different tasks across the image, video and audio modalities. We consider two variants of PolyViT. The first is PolyViT for a single modality, where we co-train three separate PolyViT models on all the tasks from either the image, video or audio modalities. The second is the multi-modal PolyViT scenario where we co-train on all nine tasks across three modalities. Here, we set $L_{adapt}$ to 0 and $L/2$ respectively to understand the effect of the number of modality-adaptor and shared layers.

We compare PolyViT to two baselines, which illustrate two alternatives to co-training. One baseline is to train 9 separate single-task models for each dataset, either ViT, ViViT or AST depending on the modality. This results in accuracies comparable to the state-of-the-art on the respective datasets, but also the largest number of total parameters. The second baseline is to use a ViT model initialized on ImageNet-21K (ViT-Im21K) and to "freeze" the encoder of the network and train only the linear output heads (Eq. 4,5) for each task. Positional embeddings, $\mathbf{p}$, and input embeddings, $\mathbf{E}$ are initialized following the methods used by ViT, ViViT or AST as described in Sec. 2.1. This baseline has the same number of parameters as PolyViT with $L_{adapt} = 0$.

Table 2 shows that PolyViT trained on a single modality achieves the highest performance on 7 of the 9 datasets. On the remaining two, the accuracy difference is negligible, as it is at most 0.3%. Moreover, the total number of parameters is 3 times less than the single-task baselines. Single-modality co-training improves accuracy the most on the smaller datasets within the modality (Kinetics 400 in video, Mini-AudioSet for audio, and CIFAR-100 for images; full dataset details in Appendix A). This suggests that co-training acts as a regularizer, as noted by Caruana (1997), that facilitates learning on smaller datasets where high-capacity models would otherwise overfit.

Table 3: Linear probing of PolyViT and single-task baselines. Similar to the protocol for evaluating self-supervised representation learning, we train only a linear classifier on top of a "frozen" transformer encoder. Note how PolyViT co-trained on all tasks transfers well to all other datasets and modalities. Models trained on audio do not transfer well to images and video, and vice versa. All models are pretrained on ImageNet-21K, and then optionally finetuned on downstream datasets.

| | | Image | | | | | | Video | | | Audio | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Model | Finetuning | C-ch101 | SUN397 | Dmlab | DTD | KITTI | PCAM | Epic K. | S-S v2 | K600 | MiT-A | K400-A |
| ViT-Im21K pretrained | – | 88.9 | 75.7 | 41.0 | 72.1 | 46.9 | 80.2 | 10.0 | 17.8 | 66.6 | 4.9 | 10.8 |
| ViT | ImageNet-1K | **91.0** | 79.3 | 45.6 | 71.9 | 52.5 | 80.7 | 12.2 | 18.5 | 67.9 | 5.3 | 12.0 |
| PolyViT | Image tasks | 90.7 | **80.0** | 45.2 | **72.5** | 53.8 | 81.2 | 12.1 | 17.9 | 67.9 | 5.3 | 11.9 |
| ViViT | MiT | 85.2 | 73.8 | 43.0 | 69.9 | **54.9** | 81.7 | 14.9 | 26.3 | 74.2 | 5.1 | 11.9 |
| PolyViT | Video tasks | 89.2 | 77.5 | **45.9** | 71.1 | 53.5 | **83.8** | 17.2 | 27.9 | **79.7** | 5.3 | 12.2 |
| AST | VGGSound | 29.0 | 7.6 | 29.8 | 34.7 | 45.1 | 79.5 | 2.9 | 4.6 | 10.6 | 9.7 | 21.7 |
| PolyViT | Audio tasks | 38.8 | 14.7 | 31.4 | 40.1 | 43.2 | 78.4 | 3.0 | 5.8 | 14.5 | **10.3** | **22.0** |
| PolyViT $L_{adapt}=0$ | All | **91.0** | 78.2 | 45.8 | 71.8 | 52.3 | 81.9 | 16.8 | 27.9 | 77.8 | 9.6 | 20.6 |
| PolyViT $L_{adapt}=L/2$ | All | 90.7 | 77.8 | 45.1 | 72.1 | 52.5 | 82.3 | **18.0** | **28.7** | 79.4 | 9.9 | 21.1 |

Multi-modal PolyViT (final two rows) achieves competitive performance whilst using substantially fewer parameters. In particular, PolyViT with all transformer encoder layers shared between modalities ($L_{adapt} = 0$) is within 1.2% of the single-task baselines across all datasets whilst using 8.3 times fewer parameters. This model also comprehensively outperforms the ViT-Im21K Linear probe baseline which has the same number of parameters. Sharing half the transformer layers between modalities ($L_{adapt} = L/2$) increases model capacity, and the model improves upon the corresponding single-task baseline on 4 datasets, whilst being at most 0.5% worse on the others. The total number of parameters is still reduced by a factor of 4.3 compared to the single-task baselines.

Our results are consistent when using the Large model backbone as shown in Appendix C.

### 4.4 Evaluating learned representations with linear probes

We now evaluate the feature representations learned by PolyViT by simply appending and training only a new linear head (Eq. 4,5) for a new task. This evaluation therefore follows the experimental setting commonly used in self-supervised learning to evaluate the quality of learned representations (Chen et al., 2020; Grill et al., 2020). Note that the new task can come from any one of the three modalities that PolyViT is trained on, since the modality-adaptor layers (if present) are modality-specific rather than task-specific.

In particular, we evaluate on a number of new image, audio and video datasets as detailed in Appendix D. For image classification, we include Caltech101, SUN397, DmLab, DTD, Kitti Distance and PatchCamelyon, which are datasets from the Visual Task Adaptation Benchmark (Zhai et al., 2019) not in our co-training set. For video classification, we also include Epic Kitchens, Something-Something v2 and Kinetics 600. Finally, for audio classification, we use the audio versions of the Moments in Time and Kinetics 400 datasets.

We use PolyViT-Base and take linear probes of all the PolyViT models from Sec. 4.3, i.e. three single-modality models and two multi-modal models trained on all tasks with $L_{adapt} = 0$ and $L_{adapt} = L/2$ respectively. Our baseline models are those not performing co-training. Namely, we use ViT trained only on ImageNet-21K (ViT-Im21K) as a baseline, followed by ViT, ViViT and AST initialized from ViT-Im21K and finetuned on ImageNet, Moments in Time and VGGSound respectively (since these are the largest datasets for each respective modality).

Table 3 shows how PolyViT trained on multiple modalities learns cross-modal feature representations that perform well on all 11 linear evaluation tasks across three different modalities (last two rows). This holds even when all the layers of the PolyViT transformer layer are shared, and thus the total number of parameters is roughly equal to a single-task model. PolyViT where the first half of the transformer encoder layers are modality-specific (final row), has more parameters and in general performs better. Furthermore, for the Epic Kitchens (video), Something-Something v2 (video) and Caltech 101 (image) datasets, multi-modal PolyViT transfers better than single-modality baselines. Table 3 thus demonstrates how co-training on multiple modalities facilitates learning powerful, transferable feature representations that can be used on multiple downstream tasks.

Table 4: State-of-the-art comparison for video classification. For ViViT, the current published state-of-the-art, we compare to numbers reported by Arnab et al. (2021) using standard dataset protocols.

|  |  |  | Kinetics 400 |  | Kinetics 600 |  | Moments in Time |  |
|---|---|---|---|---|---|---|---|---|
| Model | #Models | #Params | Top 1 | Top 5 | Top 1 | Top 5 | Top 1 | Top 5 |
| ViViT | 3 | 913M | 80.6 | 94.7 | 82.5 | **95.6** | 38.0 | 64.9 |
| PolyViT | 1 | **308M** | **82.4** | **95.0** | **82.9** | 95.5 | **38.6** | **65.5** |

Table 5: State-of-the-art comparison for audio classification. We compare to MBT, the current state-of-the-art, using the same protocols as Nagrani et al. (2021). For AudioSet, we train on the balanced subset, AS-500k, following Nagrani et al. (2021) as described in Sec. 4.1.

|  |  |  | AudioSet | VGGSound |  |
|---|---|---|---|---|---|
| Model | #Models | #Params | mAP | Top 1 | Top 5 |
| MBT (audio-only) | 2 | 172M | 44.3 | 52.3 | 78.1 |
| PolyViT | 1 | **87M** | **44.5** | **55.1** | **80.4** |

Models trained on only a single modality, as expected, do not in general learn feature representations that transfer well to other modalities. In particular, models trained on audio tasks do not transfer at all to images and videos, and vice versa. Models trained on video, however, still perform well on images, with video-trained models performing the best on the DmLab, PCAM and KITTI-Distance datasets. We believe this is due to the commonalities between the image and video modalities. Observe that in the majority of cases, single-modality PolyViT models perform better on linear probing than the corresponding single-task baselines, especially for video and audio.

### 4.5 State-of-the-art performance with single-modality co-training

Motivated by the performance of single-modality co-training in Tab. 2, we perform larger-scale co-training experiments with this method on video and audio classification.

Tables 4 and 5 show that we achieve state-of-the-art results in both of these domains whilst using also significantly fewer parameters. For video classification, we co-train PolyViT-Large with a smaller tubelet size (and hence greater number of tokens) of $2 \times 16 \times 16$ on Kinetics-400, -600 and Moments in Time. We compare to ViViT (Arnab et al., 2021) which is the current published, state-of-the-art, and uses the same initialization, transformer backbone and number of tokens. As shown in Tab. 4, we surpass the state-of-the-art on all three datasets, with the largest improvement of 1.8% being on Kinetics 400, which is also the smallest dataset. This is in line with our findings from Sec. 4.3 and shows that co-training has a regularizing effect that reduces overfitting and improves performance the most on smaller datasets. Moreover, by co-training on three datasets, we reduce the total number of parameters required by almost three times compared to separately trained ViViT models.

On audio classification, we compare to the current state-of-the-art using audio information only, MBT (Nagrani et al., 2021), using the same Base backbone and other experimental settings as the authors. As with our video experiments, we improve on both datasets (AudioSet and VGGSound), whilst using about half the total number of parameters. Once again, we observe larger improvements (2.8%) on VGGSound, which is the smaller dataset, as co-training has a regularizing effect.

## 5 Related work

Our model is related to multi-task learning and transformer models, which we discuss below.

Multi-task learning aims to develop models that can address multiple tasks whilst sharing parameters and computation between them (Caruana, 1997). In computer vision, multiple papers have developed models which predict multiple outputs (for example semantic segmentation and surface normals), given a single input image (Eigen & Fergus, 2015; Kokkinos, 2017; Zhang et al., 2014). Numerous works have also observed that although multi-task models are more versatile, their accuracies are lower than single-task models, and this accuracy deficit increases with the number of tasks, or by simultaneously performing unrelated tasks (Kokkinos, 2017; Zamir et al., 2018; McCann et al., 2018). Moreover, jointly training a network to simultaneously perform multiple tasks has typically required careful calibration of the individual tasks, to ensure that none of the task-specific losses dominates another. Methods to mitigate this include gradient-normalization (Chen et al., 2018) and -surgery (Yu et al., 2020) and adaptive loss weights (Sener & Koltun, 2018; Kendall et al., 2018).

Our work differs in that although our network is capable of performing multiple tasks, it performs one task at a time for a given input. Note that this setting is also more suited to the case of handling multiple input modalities. Such an approach was also performed by Maninis et al. (2019) who named it "single-tasking of multiple tasks" in the context of computer vision. However, in natural language processing (NLP), this setting is still referred to as "multi-task learning" (Collobert & Weston, 2008). Furthermore, our co-training strategy is simple, and alternates between performing SGD for batches of separate tasks. For high-capacity transformer models, we find that co-training on multiple datasets simultaneously helps to regularize the model on a dataset that it would otherwise overfit on, thus achieving accuracy improvements from co-training. Previous works have improved performance on additional tasks only by introducing extra task-specific parameters (Misra et al., 2016; Houlsby et al., 2019) which are typically conditioned on the input (Rebuffi et al., 2017; Maninis et al., 2019).

We also note that simlar co-training setups to our work have been explored in NLP. A recent paradigm in NLP has been to reduce different tasks to a common, unified framework (Raffel et al., 2019; Brown et al., 2020; McCann et al., 2018). This common interface allows co-training a single model to perform multiple tasks, as it effectively involves concatenating multiple datasets together (Raffel et al., 2019; Khashabi et al., 2020; Tay et al., 2020).

Although the majority of previous multi-task learning works have considered only a single modality, Kaiser et al. (2017) presented an early effort on multi-modal models. Their heterogeneous model consisted of convolutional layers to process images, and attention and mixture-of-experts layers to model text. Their results, however, were not competitive with the state-of-the-art as ours.

Our model, motivated by Dosovitskiy et al. (2021), can readily handle diverse modalities, as transformers operate on any sequence of tokens. Relevant to us, the Perceiver (Jaegle et al., 2021) is a transformer architecture that can process different modalities. Instead of tokenizing images or audio spectrograms with non-overlapping patches like Dosovitskiy et al. (2021) and Gong et al. (2021) respectively, Jaegle et al. (2021) operate directly on the raw input by projecting it into a smaller, latent set of tokens using cross-attention. Although this architecture is capable of processing different modalities, the authors train separate networks with separate parameters for each task. Therefore, they do not consider co-training scenarios like our work. MBT (Nagrani et al., 2021), on the other hand, proposes a transformer model to fuse different modalities (for example audio and rgb frames of videos) to solve a single task. Once again, separate model parameters are used for each task.

Hu & Singh (2021) co-train a transformer-based model, but specifically for vision-and-language tasks. The authors use an encoder-decoder architecture (Vaswani et al., 2017), where only the decoder is shared among different tasks, and the encoder is specialized for each modality. In particular, the visual encoder is DeTR (Carion et al., 2020) and the text encoder is BERT (Devlin et al., 2019), and each component is pretrained separately. In contrast to our work, they do not consider scenarios where the entire transformer backbone is shared among different tasks, nor do they thoroughly analyze how to co-train multiple tasks and modalities like our work. Furthermore, their approach does not outperform single-task baselines as our work does. Other papers concentrating on multi-task learning of vision-and-language tasks include (Lu et al., 2019; Li et al., 2020; Lu et al., 2020).

Finally, we note that Akbari et al. (2021) have recently processed multiple modalities with a single transformer backbone for cross-modal, contrastive self-supervised learning as previously done by (Alayrac et al., 2020; Miech et al., 2020) with convolutional models. The focus of these works is pretraining models to learn powerful representations, and is therefore an alternative to the supervised pretrained on large datasets like ImageNet-21K that we used.

## 6 Conclusion and Future Work

By co-training PolyViT on a single modality, we have achieved state-of-the-art results on three video and two audio datasets while reducing the total number of parameters linearly compared to single-task models. PolyViT co-trained on multiple modalities is even more parameter-efficient, still competitive with the state-of-the-art, and learns feature representations that generalize across multiple modalities enabling us to learn new tasks by simpling learning an additional output head. Co-training is also practical, as we don't need to tune hyperparameters from the joint space of datasets, but can simply re-use training hyperparameters from single-task models. Moreover, we can achieve accuracy improvements from training for the same number of total steps. Future work is to co-train on large-scale pretraining datasets, and consider additional modalities, like text.

### Ethics Statement

Our work presents a method for performing image-, audio- and video-classification with a single parameter-efficient model. Classification of perceptual data (images, audio and video) is a general technology with a wide range of potential applications. While we are unaware of all potential applications, it is important to be aware that each application has its own merits and societal implications depending on the intentions of the individuals building and using the system. We also note that training datasets contain biases that may render models trained on them unsuitable for certain applications. It is possible that people use classification models (intentionally or not) to make decisions that impact different groups in society differently.

### Reproducibility Statement

We have included exhaustive descriptions of our datasets, training hyperparameters and experimental details in the main paper and appendices. Note that we have included appendices to provide details for each of our main experiments. For example, Appendix B provides more details for the experiments in Sec. 4.2, Appendix C for experiments in Sec. 4.3 up to Appendix E for experiments in Sec. 4.5. For additional clarity, the appendices contain 5 tables detailing the configurations for each of our experiments. We have also used only publicly available datasets in all of our experiments. Finally, we will release code and models upon acceptance.

## References

Hassan Akbari, Linagzhe Yuan, Rui Qian, Wei-Hong Chuang, Shih-Fu Chang, Yin Cui, and Boqing Gong. Vatt: Transformers for multimodal self-supervised learning from raw video, audio and text. In *ICCV*, 2021.

Jean-Baptiste Alayrac, Adrià Recasens, Rosalia Schneider, Relja Arandjelović, Jason Ramapuram, Jeffrey De Fauw, Lucas Smaira, Sander Dieleman, and Andrew Zisserman. Self-supervised multimodal versatile networks. In *NeurIPS*, 2020.

Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lucic, and Cordelia Schmid. Vivit: A video vision transformer. In *ICCV*, 2021.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *arXiv preprint arXiv:2005.14165*, 2020.

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020.

Rich Caruana. Multitask learning. *Machine learning*, 1997.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020.

Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *ICML*, 2018.

Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *ICML*, 2008.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2019.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.

David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *ICCV*, 2015.

Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 1999.

Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

Yuan Gong, Yu-An Chung, and James Glass. AST: Audio Spectrogram Transformer. In *Proc. Interspeech*, 2021.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent: A new approach to self-supervised learning. In *NeurIPS*, 2020.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017.

Dan Hendrycks and Kevin Gimpel. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *arXiv preprint arXiv:1606.08415*, 2016.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *ICML*, 2019.

Ronghang Hu and Amanpreet Singh. Unit: Multimodal multitask learning with a unified transformer. In *ICCV*, 2021.

Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q. Weinberger. Deep networks with stochastic depth. In *ECCV*, 2016.

Andrew Jaegle, Felix Gimeno, Andrew Brock, Andrew Zisserman, Oriol Vinyals, and Joao Carreira. Perceiver: General perception with iterative attention. In *arXiv preprint arXiv:2103.03206*, 2021.

Lukasz Kaiser, Aidan N Gomez, Noam Shazeer, Ashish Vaswani, Niki Parmar, Llion Jones, and Jakob Uszkoreit. One model to learn them all. In *arXiv preprint arXiv:1706.05137*, 2017.

Will Kay, João Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman, and Andrew Zisserman. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017.

Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *CVPR*, 2018.

Daniel Khashabi, Sewon Min, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. Unifiedqa: Crossing format boundaries with a single qa system. *arXiv preprint arXiv:2005.00700*, 2020.

Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. In *NeurIPS*, 2017.

Iasonas Kokkinos. Ubernet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory. In *CVPR*, 2017.

Gen Li, Nan Duan, Yuejian Fang, Ming Gong, and Daxin Jiang. Unicoder-vl: A universal encoder for vision and language by cross-modal pre-training. In *AAAI*, 2020.

Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In *NeurIPS*, 2019.

Jiasen Lu, Vedanuj Goswami, Marcus Rohrbach, Devi Parikh, and Stefan Lee. 12-in-1: Multi-task vision and language representation learning. In *CVPR*, 2020.

Kevis-Kokitsi Maninis, Ilija Radosavovic, and Iasonas Kokkinos. Attentive single-tasking of multiple tasks. In *CVPR*, 2019.

Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. The natural language decathlon: Multitask learning as question answering. *arXiv preprint arXiv:1806.08730*, 2018.

Antoine Miech, Jean-Baptiste Alayrac, Lucas Smaira, Ivan Laptev, Josef Sivic, and Andrew Zisserman. End-to-end learning of visual representations from uncurated instructional videos. In *CVPR*, 2020.

Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. Cross-stitch networks for multi-task learning. In *CVPR*, 2016.

Arsha Nagrani, Shan Yang, Anurag Arnab, Aren Jansen, Cordelia Schmid, and Chen Sun. Attention bottlenecks for multimodal fusion. *arXiv preprint arXiv:2107.00135*, 2021.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2019.

Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Learning multiple visual domains with residual adapters. In *NeurIPS*, 2017.

Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. In *NeurIPS*, 2018.

Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. *ICCV*, 2017.

Yi Tay, Zhe Zhao, Dara Bahri, Donald Metzler, and Da-Cheng Juan. Hypergrid transformers: Towards a single model for multiple tasks. In *ICLR*, 2020.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. In *NeurIPS*, 2020.

Amir R. Zamir, Alexander Sax, William Shen, Leonidas J. Guibas, Jitendra Malik, and Silvio Savarese. Taskonomy: Disentangling task transfer learning. In *CVPR*, 2018.

Xiaohua Zhai, Joan Puigcerver, Alexander Kolesnikov, Pierre Ruyssen, Carlos Riquelme, Mario Lucic, Josip Djolonga, Andre Susano Pinto, Maxim Neumann, Alexey Dosovitskiy, et al. A large-scale study of representation learning with the visual task adaptation benchmark. *arXiv preprint arXiv:1910.04867*, 2019.

Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. Mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

Zhanpeng Zhang, Ping Luo, Chen Change Loy, and Xiaoou Tang. Facial landmark detection by deep multi-task learning. In *ECCV*, 2014.

# A Experimental set-up: additional details

**Task details and input dimensions.** See Tables 6 and 7. For each task, the number of linear warmup steps is set as reported in (Dosovitskiy et al., 2021; Arnab et al., 2021; Nagrani et al., 2021). When co-training, we simply use the sum of all warmup steps for each co-trained task. Similarly to (Dosovitskiy et al., 2021), we select the best learning rate on a set $\{0.03, 0.1, 0.3\}$ using the validation score. For video and audio datasets, we reuse learning rates reported in (Arnab et al., 2021) and (Nagrani et al., 2021) respectively. As in (Arnab et al., 2021; Nagrani et al., 2021), we use zero initialization for output head kernels $\mathbf{W}_{out}$. For image datasets, on a single-task evaluation, we find that LeCun normal $\mathbf{W}_{out}$ initializer (Klambauer et al., 2017) works best. For the “ViT-Im21K linear probe” baseline, we use the same training procedure as for single-task baselines, with the difference that 1) only the head parameters are updated and 2) on image tasks, we run separate learning rate grid searches on the set $\{0.03, 0.1, 0.3\}$.

**Train, validation and test splits.** Similarly to (Dosovitskiy et al., 2021), we take 2% of CIFAR 10/100 train sets for validation, 10% of Pets train set for validation and 1% of ImageNet-1k train set for validation. We use standard test sets for these datasets. For RESISC45, we use 20% of the train set for validation and 20% for testing. We use standard train, validation and test sets for video and audio tasks.

**Augmentation and regularization.** We don’t use augmentation for image tasks. We do video and audio preprocessing and augmentation as done in (Arnab et al., 2021; Nagrani et al., 2021) respectively. For audio tasks, as in (Nagrani et al., 2021), we use Mixup (Zhang et al., 2017) with $\alpha = 0.3$ and stochastic depth regularization (Huang et al., 2016) with $p = 0.3$. Stochastic depth is applied along both audio adaptor and shared layers.

Table 6: Experimental set-up: tasks and their properties. For image tasks, the indicated learning rates are obtained by a grid search over $\{0.03, 0.1, 0.3\}$ on single-task baselines using the validation set accuracy. These values are used for single-task baselines and for PolyViT variants.

| Dataset | Abbre-viation | Moda-lity | Clas-ses | Train size | Train steps | Learning rate | Warmup steps | $\mathbf{W}_{out}$ init |
|---|---|---|---|---|---|---|---|---|
| CIFAR 100 | C100 | Image | 100 | 50.0K | 10K | 0.03 | 500 | LeCun normal |
| CIFAR 10 | C10 | Image | 10 | 50.0K | 10K | 0.03 | 500 | LeCun normal |
| Oxford-IIIT Pets | Pets | Image | 37 | 3.68K | 500 | 0.03 | 100 | LeCun normal |
| RESISC45 | R45 | Image | 45 | 31.5K | 2.5K | 0.1 | 200 | LeCun normal |
| ImageNet-1k | Im1K | Image | 1000 | 1.28M | 20K | 0.03 | 500 | LeCun normal |
| Kinetics 400 | K400 | Video | 400 | 215K | 100.7K<br>(30 epochs) | 0.1 | 2.5 epochs | Zeros |
| Moments in Time | MiT | Video | 339 | 791K | 123.6K<br>(10 epochs) | 0.25 | 2.5 epochs | Zeros |
| Mini-Audioset | MiniAS | Audio | 527 | 20.4K | 15.9K<br>(50 epochs) | 0.5 | 2.5 epochs | Zeros |
| VGGSound | VGG | Audio | 309 | 172K | 135K<br>(50 epochs) | 0.5 | 2.5 epochs | Zeros |

# B Selecting the best task sampling schedule: additional experimental details

For the accumulating schedule, we set learning rate to the smallest value across tasks (0.03). We draw a random task order for the Task-by-task schedule, which is as follows: C100 → MiT → K400 → MiniAS → VGG → Pets → C10 → Im1K → R45.

Table 7: Input dimensions for different modalities. Sequence length is computed as $1 + [(T/t)\times](H/h) \times (W/w)$ (one class token and patch tokens). Note that for shared transformer layers, we reuse the same parameters for sequences of different lengths.

| Modality | Input size, $[T\times]H \times W$ | Patch size, $[t\times]h \times w$ | Sequence length | Batch size |
|---|---|---|---|---|
| Image (pretraining) | 224 × 224 | 16 × 16 | 197 | 4096 |
| Image | 384 × 384 | 16 × 16 | 577 | 512 |
| Video | 32 × 224 × 224 | 4 × 16 × 16 | 1569 | 64 |
| Audio (spectrogram) | 800 × 128 | 16 × 16 | 401 | 64 |

## C Co-training with PolyViT: additional experimental details and results

**Evaluation on video and audio tasks.** To get test performance on video and audio tasks, we perform multiple-crop evaluation as described in (Arnab et al., 2021; Nagrani et al., 2021) for videos and audio respectively.

**Results for the Large configuration.** See Table 8. Since (Nagrani et al., 2021) don't report results on a Large configuration, for audio tasks we do an additional hyperparameter tuning for single-task baselines on validation sets. As a result, we use Mixup $\alpha = 0.5, 0.7$ for MiniAS and VGGSound respectively. Also, we use 30 epochs for MiniAS instead of 50 for the Base model. In addition, we run separate learning rate grid searches for all image tasks, separately for single-task baselines and ViT-Im21K linear probes. We apply all mentioned hyperparameter changes, obtained for the single-task baselines, to all PolyViT runs. In all other aspects, Large set-up is the same as Base.

Table 8: Co-training with PolyViT, Large model configuration. Test accuracy (%) and mAP (for MiniAS, %) are reported. As indicated by the "# models" column, some rows correspond to multiple models, then the total number of parameters is computed across all models.

| | | | Image | | | | | Video | | Audio | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Model | #Models | #Params | C100 | C10 | Pets | R45 | Im1K | K400 | MiT | MiniAS | VGG |
| ViT-Im21k Linear probe | 1 | **312M** | 84.4 | 95.6 | 91.8 | 89.2 | 82.6 | 67.7 | 26.8 | 12.8 | 19.1 |
| Single-task baseline | 9 | 3033M | 93.3 | 99.2 | 94.8 | **97.3** | **85.1** | 79.6 | 37.1 | 30.0 | **51.8** |
| PolyViT, 1 modality | 3 | 917M | **93.9** | **99.4** | **95.5** | 96.9 | **85.1** | 80.6 | **38.8** | **37.9** | 50.7 |
| PolyViT, $L_{adapt} = 0$ | 1 | **312M** | 91.4 | 99.0 | 94.7 | 96.8 | 82.6 | 78.9 | 35.8 | 33.3 | 49.9 |
| PolyViT, $L_{adapt} = L/2$ | 1 | 615M | 91.1 | 99.1 | 95.0 | 97.0 | 82.8 | **81.0** | 37.7 | 34.1 | 50.4 |

## D Linear probes: additional experimental details

**Task details.** See Table 9. For linear probes, we use the same input dimensions as reported in Table 7. For image tasks, we reuse the number of train and warmup steps from the RESISC45 task (Table 6). For video and audio tasks, we used hyperparameters reported in (Arnab et al., 2021) and (Nagrani et al., 2021) respectively, with the difference that we only optimize output head parameters during training. As for the co-training setup, we use multiple-crop evaluation on video and audio tasks.

**Train, validation and test splits.** For image tasks, we use 2% of the train set as a validation set and standard test sets. We use standard train, validation and test sets for video and audio tasks.

**Converting patch and positional embeddings for cross-modal probes.** In order to take linear probes of image-only models (ViT and PolyViT trained on images) on audio tasks (and vice versa), we leave patch embeddings as they are and 2D-interpolate positional embeddings to the correct resolution. When taking linear probes of video-only models on image or audio tasks, in order to obtain 16 × 16 patch embeddings, we take a sum along the first (frame) axis of 3D video patch embeddings of shape $4 \times 16 \times 16$. In order to adapt positional embeddings, we take a mean value of positional embeddings for each frame, and then 2D-interpolate the result to the correct resolution. When taking linear probes of image- or audio-only models on video tasks, we repeat 2D patch embeddings along the frame axis in order to obtain 3D patch embeddings. We also 2D-interpolate positional embeddings to the frame resolution and repeat them for each frame.

**Augmentation and regularization.** We don't use augmentation for image tasks. We do video and audio preprocessing and augmentation as done in (Arnab et al., 2021; Nagrani et al., 2021) respectively. As in (Arnab et al., 2021), we use Mixup (Zhang et al., 2017) with $\alpha = 0.3$ for the S-S v2 task.

Table 9: Tasks used for linear probes. Indicated learning rate grid search is done for all models using validation set performance.

| Dataset (task) | Abbreviation | Modality | Train steps | Learning rate | Warmup steps | $\mathbf{W}_{out}$ init |
|---|---|---|---|---|---|---|
| Caltech101 | C-ch101 | Image | 2.5K | Grid search, $\{0.03, 0.1, 0.3\}$ | 200 | LeCun normal |
| SUN397 | SUN397 | Image | 2.5K | Grid search, $\{0.03, 0.1, 0.3\}$ | 200 | LeCun normal |
| Dmlab | Dmlab | Image | 2.5K | Grid search, $\{0.03, 0.1, 0.3\}$ | 200 | LeCun normal |
| DTD | DTD | Image | 2.5K | Grid search, $\{0.03, 0.1, 0.3\}$ | 200 | LeCun normal |
| KITTI Distance | KITTI | Image | 2.5K | Grid search, $\{0.03, 0.1, 0.3\}$ | 200 | LeCun normal |
| PatchCamelyon | PCAM | Image | 2.5K | Grid search, $\{0.03, 0.1, 0.3\}$ | 200 | LeCun normal |
| Epic Kitchens | Epic K. | Video | 30 epochs | 0.5 | 2.5 epochs | Zeros |
| Something-Something v2 | S-S v2 | Video | 35 epochs | 0.4 | 2.5 epochs | Zeros |
| Kinetics 600 | K600 | Video | 30 epochs | 0.1 | 2.5 epochs | Zeros |
| Moments in Time (audio) | MiT-A | Audio | 10 epochs | 0.5 | 2.5 epochs | Zeros |
| Kinetics 400 (audio) | K400-A | Audio | 30 epochs | 0.5 | 2.5 epochs | Zeros |

# E State-of-the-art performance on one modality: additional experimental details

For the PolyViT experiment on the video modality, we reuse hyperparameters reported in (Arnab et al., 2021) for Kinetics 400/600 and Moments in Time. See Table 10 for the dataset details and exact hyperparameters used during the experiment. These hyperparameters coincide with those reported in Table 6 for Kinetics 400 and Moments in Time and in Table 9 for Kinetics 600. The only difference is that we use a more granular 3D patch size ($2 \times 16 \times 16$) and Large model configuration.

For the audio experiment, similarly, we reuse all hyperparameters reported in (Nagrani et al., 2021) for AS-500k and VGGSound experiments (audio-only). See Table 11 for the dataset details and exact hyperparameters used for the experiment. These hyperparameters almost coincide with those reported in Table 9 with a change MiniAS $\rightarrow$ AS-500k. The only exception is that we use 30 epochs and Mixup $\alpha = 0.5$ for AS-500k.

Table 10: Set-up for the co-training on videos. Train steps and warmup steps are summed to get the number of train and warmup steps during co-training as we use the "Weighted" task sampling method.

| Dataset | Moda-<br>lity | Clas-<br>ses | Train<br>size | Train<br>steps | Batch<br>size | Learning<br>rate | Warmup<br>steps | $\mathbf{W}_{out}$<br>init |
|---|---|---|---|---|---|---|---|---|
| Kinetics 400 | Video | 400 | 215K | 101K<br>(30 epochs) | 64 | 0.1 | 2.5 epochs | Zeros |
| Kinetics 600 | Video | 600 | 363K | 170K<br>(30 epochs) | 64 | 0.1 | 2.5 epochs | Zeros |
| Moments in Time | Video | 339 | 791K | 123.6K<br>(10 epochs) | 64 | 0.25 | 2.5 epochs | Zeros |

Table 11: Set-up for the co-training on audio. Train steps and warmup steps are summed to get the number of train and warmup steps during co-training as we use the "Weighted" task sampling method.

| Dataset | Moda-<br>lity | Clas-<br>ses | Train<br>size | Train<br>steps | Mixup | Batch<br>size | Learning<br>rate | Warmup<br>steps | $\mathbf{W}_{out}$<br>init |
|---|---|---|---|---|---|---|---|---|---|
| AS-500k | Audio | 527 | 509K | 239K<br>(30 epochs) | 0.5 | 64 | 0.5 | 2.5 epochs | Zeros |
| VGGSound | Audio | 309 | 172K | 135K<br>(50 epochs) | 0.3 | 64 | 0.5 | 2.5 epochs | Zeros |