What Makes Training Multi-modal Classification Networks Hard?

Weiyao Wang, Du Tran, Matt Feiszli
Facebook AI
{weiyao,tranu,mdf}@fb.com

Abstract

Consider end-to-end training of a multi-modal vs. a uni-modal network on a task with multiple input modalities: the multi-modal network receives more information, so it should match or outperform its uni-modal counterpart. In our experiments, however, we observe the opposite: the best uni-modal network often outperforms the multi-modal network. This observation is consistent across different combinations of modalities and on different tasks and benchmarks for video classification.

This paper identifies two main causes for this performance drop: first, multi-modal networks are often prone to overfitting due to their increased capacity. Second, different modalities overfit and generalize at different rates, so training them jointly with a single optimization strategy is sub-optimal. We address these two problems with a technique we call Gradient-Blending, which computes an optimal blending of modalities based on their overfitting behaviors. We demonstrate that Gradient Blending outperforms widely-used baselines for avoiding overfitting and achieves state-of-the-art accuracy on various tasks including human action recognition, ego-centric action recognition, and acoustic event detection.

1. Introduction

Consider a late-fusion multi-modal network, trained end-to-end to solve a task. Uni-modal solutions are a strict subset of the solutions available to the multi-modal network; a well-optimized multi-modal model should, in theory, always outperform the best uni-modal model. However, we show here that current techniques do not always achieve this. In fact, what we observe is contrary to common sense: the best uni-modal model often outperforms the joint model, across different modalities (Table 1) and datasets (details in section 3). Anecdotally, the performance drop with multiple input streams appears to be common and was noted in [24, 3, 38, 44]. This (surprising) phenomenon warrants investigation and solution.

Upon inspection, the problem appears to be overfitting:

- **Table 1:** Uni-modal networks consistently outperform multi-modal networks. Best uni-modal networks vs late fusion multi-modal networks on Kinetics using video top-1 validation accuracy. Single stream modalities include video clips (RGB), Optical Flow (OF), and Audio (A). Multi-modal networks use the same architectures as uni-modal, with late fusion by concatenation at the last layer before prediction.

Figure 1: Standard regularizers do not provide a good improvement over the best Uni-modal network. Best uni-modal network (RGB) vs standard approaches on a multi-modal network (RGB+Audio) on Kinetics. Various methods to avoid overfitting (orange: Pre-training, Early-stopping, and Dropout) do not solve the issue. Different fusion architectures (red: Mid-concatenation fusion, SE-gate, and NL-gate) also do not help. Dropout and Mid-concatenation fusion approaches provide small improvements (+0.3% and +0.2%), while other methods degrade accuracy.

multi-modal networks have higher train accuracy and lower validation accuracy. Late fusion audio-visual (A+RGB) network has nearly two times the parameters of a visual network, and one may suspect that the overfitting is caused by the increased number of parameters.

There are two ways to approach this problem. First, one can consider solutions such as dropout [43], pre-training, or early stopping to reduce overfitting. On the other hand, one may speculate that this is an architectural deficiency. We experiment with mid-level fusion by concatenation [37] and fusion by gating [31], trying both Squeeze-and-Excitation (SE) [26] and Non-Local (NL) [51] gates.

Remarkably, none of these provide an effective solu-
tion. For each method, we record the best audio-visual results on Kinetics in Figure 1. Pre-training fails to offer improvements, and early stopping tends to under-fit the RGB stream. Mid-concat and dropout provide only modest improvements over RGB model. We note that dropout and mid-concat (with 37% fewer parameters compared to late-concat) make 1.5% and 1.4% improvements over late-concat, confirming the overfitting problem with late-concat. We refer to supplementary materials for details.

How do we reconcile these experiments with previous multi-modal successes? Multi-modal networks have successfully been trained jointly on tasks including sound localization [59], image-audio alignment [5], and audio-visual synchronization [37, 34]. However, these tasks cannot be performed with a single modality, so there is no unimodal baseline and the performance drop found in this paper does not apply. In other work, joint training is avoided entirely by using pre-trained uni-modal features. Good examples include two-stream networks for video classification [41, 49, 19, 12] and image+text classification [6, 31]. These methods do not train multiple modalities jointly, so they are again not comparable, and their accuracy may likely be sub-optimal due to independent training.

Our contributions in this paper include:

- We empirically demonstrate the significance of overfitting in joint training of multi-modal networks, and we identify two causes for the problem. We show the problem is architecture agnostic: different fusion techniques can also suffer the same overfitting problem.
- We propose a metric to understand the problem quantitatively: the overfitting-to-generalization ratio (OGR), with both theoretical and empirical justification.
- We propose a new training scheme which minimizes OGR via an optimal blend (in a sense we make precise below) of multiple supervision signals. This Gradient-Blending (G-Blend) method gives significant gains in ablations and achieves state-of-the-art (SoTA) accuracy on benchmarks including Kinetics, EPIC-Kitchen, and AudioSet by combining audio and visual signals.

We note that G-Blend is task-agnostic, architecture-agnostic and applicable to other scenarios (e.g. used in [39] to combine point cloud with RGB for 3D object detection)

1.1. Related Work

Video classification. Video understanding has been one of the most active research areas in computer vision recently. There are two unique features with respect to videos: temporal information and multi-modality. Previous works have made significant progress in understanding temporal information [27, 45, 50, 40, 47, 55, 17]. However, videos are also rich in multiple modalities: RGB frames, motion vectors (optical flow), and audio. Previous works that exploit the multi-modal natures primarily focus on RGB+Optical Flow, with the creation of two-stream fusion networks [41, 19, 18, 49, 12], typically using pre-trained features and focusing on the fusion [27, 19] or aggregation architectures [57]. In contrast, we focus on joint training of the entire network. Instead of focusing on the architectural problem, we study model optimization: how to jointly learn and optimally blend multi-modal signals. With proper optimization, we show audio is useful for video classification.

Multi-modal networks. Our work is related to previous research on multi-modal networks [7] for classifications [41, 49, 19, 21, 12, 6, 10, 31], which primarily uses pre-training in contrast to our joint training. On the other hand, our work is related to cross-modal tasks [54, 20, 42, 4, 58, 24, 9] and cross-modal self-supervised learning [59, 5, 37, 34]. These tasks either take one modality as input and make prediction on the other modality (e.g. Visual-Q&A [4, 58, 24], image captioning [9], sound localization [37, 59] in videos) or uses cross-modality correspondences as self-supervision (e.g. image-audio correspondence [5], video-audio synchronization [34]). Instead, we try to address the problem of joint training of multi-modal networks for classification.

Multi-task learning. Our proposed Gradient-Blending training scheme is related to previous works in multi-task learning in using auxiliary loss [33, 16, 30, 13]. These methods either use uniform/manually tuned weights, or learn the weights as parameters during training (no notion of overfitting prior used), while our work re-calibrates supervision signals using a prior OGR.

2. Multi-modal training via Gradient-Blending

2.1. Background

Uni-modal network. Given train set $\mathcal{T} = \{X_1,...,X_n, y_1,...,y_n\}$, where $X_i$ is the $i$-th training example and $y_i$ is its true label, training on a single modality $m$ (e.g. RGB frames, audio, or optical flows) means minimizing an empirical loss:

$$L(C(\varphi_m(X)), y)$$

where $\varphi_m$ is normally a deep network with parameter $\Theta_m$, and $C$ is a classifier, typically one or more fully-connected (FC) layers with parameter $\Theta_C$. For classification problems considered here, $L$ is the cross entropy loss. Minimizing Eq. 1 gives a solution $\Theta_m^*$ and $\Theta_C^*$. Fig. 2a shows independent training of two modalities $m_1$ and $m_2$.

Multi-modal network. We train a late-fusion model on $M$ different modalities ($\{m_i\}_{i=1}^M$). Each modality is processed by a different deep network $\varphi_{m_i}$, with parameter $\Theta_{m_i}$, and their features are fused and passed to a classifier $C$. Formally, training is done by minimizing the loss:

$$L_{\text{multi}} = L(C(\varphi_{m_1} \oplus \varphi_{m_2} \oplus \cdots \oplus \varphi_{m_M}), y)$$

where $\oplus$ denotes a fusion operation (e.g. concatenation). Fig. 2b shows an example of a joint training of two modalities $m_1$ and $m_2$. The multi-modal network in Eq. 2 is a
Figure 2: Uni- vs. multi-modal joint training. a) Uni-modal training of two different modalities. b) Naive joint training of two modalities by late fusion. c) Joint training of two modalities with weighted blending of supervision signals. Different deep network encoders (white trapezoids) produce features (blue or pink rectangles) which are concatenated and passed to a classifier (yellow rounded rectangles).

Figure 3: Overfitting-to-Generalization Ratio. Between any two training checkpoints, we can measure the change in overfitting and generalization. When $\Delta N$ is small, the network is learning well and not overfitting much.

super-set of the uni-model network in Eq. 1: for any solution to Eq. 1 on any modality $m_i$, one can construct an equally-good solution to Eq. 2 by choosing parameters $\Theta_i$ that mute all modalities other than $m_i$. In practice, this solution is not found, and we next explain why.

2.2. Generalizing vs. Overfitting

Overfitting is typically understood as learning patterns in a train set that do not generalize to the target distribution. Given model parameters at epoch $N$, let $\mathcal{L}_N^*$ be the model’s average loss over the fixed train set, and $\mathcal{L}_N'$ be the “true” loss w.r.t the hypothetical target distribution. (In what follows, $\mathcal{L}_N^*$ is approximated by a held-out validation loss $\mathcal{L}_N^V$.) We define overfitting at epoch $N$ as the gap between $\mathcal{L}_N^*$ and $\mathcal{L}_N'$ (approximated by $O_N$ in fig. 3). The quality of training between two model checkpoints can be measured by the change in overfitting and generalization ($\Delta O, \Delta G$ in fig. 3). Between checkpoints $N$ and $N + n$, we can define the overfitting-to-generalization-ratio ($OGR$):

$$OGR = \frac{\Delta O_{N,n}}{\Delta G_{N,n}} = \frac{O_{N+n} - O_N}{\mathcal{L}_N^* - \mathcal{L}_{N+n}^*}$$

$OGR$ between checkpoints measures the quality of learned information (with cross-entropy loss, it is the ratio of bits not generalizable to bits which do generalize). We propose minimizing $OGR$ during training. However, optimizing $OGR$ globally would be very expensive (e.g. variational methods over the whole optimization trajectory). In addition, very underfit models, for example, may still score quite well (difference of train loss and validation loss is very small for underfitting models; in other words, $O$ is small).

Therefore, we propose to solve an infinitesimal problem: given several estimates of the gradient, blend them to minimize an infinitesimal $OGR^2$. We apply this blend to our optimization process (e.g. SGD with momentum). Each gradient step now increases generalization error as little as possible per unit gain on the validation loss, minimizing overfitting. In a multi-modal setting, this means we combine gradient estimates from multiple modalities and minimize $OGR^2$ to ensure each gradient step now produces a gain no worse than that of the single best modality. As we will see in this paper, this $L^2$ problem admits a simple, closed-form solution, is easy to implement, and works well in practice.

Consider a single parameter update step with estimate $\hat{g}$ for the gradient. As the distance between two checkpoints is small (in the neighborhood in which a gradient step is guaranteed to decrease the train loss), we use the first-order approximations: $\Delta G \approx \langle \nabla \mathcal{L}^*, \hat{g} \rangle$ and $\Delta O \approx \langle \nabla \mathcal{L}^T - \mathcal{L}^*, \hat{g} \rangle$. Thus, $OGR^2$ for a single vector $\hat{g}$ is

$$OGR^2 = \left( \frac{\langle \nabla \mathcal{L}^T - \mathcal{L}^*, \hat{g} \rangle}{\langle \nabla \mathcal{L}^*, \hat{g} \rangle} \right)^2$$

See supplementary materials for details on $OGR$.

2.3. Blending of Multiple Supervision Signals by OGR Minimization

We can obtain multiple estimates of gradient by attaching classifiers to each modality’s features and to the fused features (see fig 2c). Per-modality gradient $\{\hat{g}_i\}_{i=1}^M$ are obtained by back-propagating through each loss separately (so per-modality gradients contain many zeros in other parts of the network). Our next result allows us to blend them all into a single vector with better generalization behavior.

**Proposition 1 (Optimal Gradient Blend).** Let $\{v_k\}_{k=0}^M$ be a set of estimates for $\nabla \mathcal{L}^*$ whose overfitting satisfies $\mathbb{E} \left[ \langle \nabla \mathcal{L}^T - \nabla \mathcal{L}^*, v_k \rangle / \langle \nabla \mathcal{L}^T - \nabla \mathcal{L}^*, v_j \rangle \right] = 0$ for $j \neq k$. Given the constraint $\sum_k w_k = 1$ the optimal weights $w_k \in \mathbb{R}$ for the problem

$$w^* = \arg \min_w \mathbb{E} \left[ \left( \frac{\langle \nabla \mathcal{L}^T - \nabla \mathcal{L}^*, \sum_k w_k v_k \rangle}{\langle \nabla \mathcal{L}^*, \sum_k w_k v_k \rangle} \right)^2 \right]$$
are given by

\[ w_k^* = \frac{1}{Z} \frac{\langle \nabla L^*, v_k \rangle}{\sigma_k^2} \]  \hspace{1cm} (6)

where \( \sigma_k^2 \equiv \mathbb{E}[(\nabla L^T - \nabla L^*, y_k)^2] \) and \( Z = \sum_k \frac{\langle \nabla L^*, v_k \rangle}{\sigma_k^2} \) is a normalizing constant.

Assumption \( \mathbb{E}[(\nabla L^T - \nabla L^*, y_k)(\nabla L^T - \nabla L^*, y_j)] = 0 \) will be false when two models’ overfitting is very correlated. However, if this is the case then very little can be gained by blending their gradients. In informal experiments we have indeed observed that these cross terms are often small relative to the \( \mathbb{E}[(\nabla L^T - \nabla L^*, y_k)^2] \). This is likely due to complementary information across modalities, and we speculate that this happens naturally as joint training tries to learn complementary features across neurons. Please see supplementary materials for proof of Proposition 1, including formulas for the correlated case.

Proposition 1 may be compared with well-known results for blending multiple estimators; e.g. for the mean, a minimum-variance estimator is obtained by blending uncorrelated estimators with weights inversely proportional to the individual variances (see e.g. [1]). Proposition 1 is similar, where variance is replaced by \( O^2 \) and weights are inversely proportional to the individual \( O^2 \) (now with a numerator \( O \)).

2.4. Use of OGR and Gradient-Blending in practice

We adapt a multi-task architecture to construct an approximate solution to the optimization above (fig 2c).

**Optimal blending by loss re-weighting** At each back-propagation step, the per-modality gradient for \( m_i \) is \( \nabla L_i \), and the gradient from the fused loss is given by Eq. 2 (denote as \( \nabla L_{k+1} \)). Taking the gradient of the blended loss

\[ L_{blend} = \sum_{i=1}^{k+1} w_i L_i \]  \hspace{1cm} (7)

thus produces the blended gradient \( \sum_{i=1}^{k+1} w_i \nabla L_i \). For appropriate choices of \( w_i \), this yields a convenient way to implement gradient blending. Intuitively, loss reweighting recalibrates the learning schedule to balance the generalization/overfitting rate of different modalities.

**Measuring OGR in practice.** In practice, \( \nabla L^* \) is not available. To measure OGR, we hold out a subset \( V \) of the training set to approximate the true distribution (i.e. \( L^V \approx L^* \)). We find it is equally effective to replace the loss measure by an accuracy metric to compute \( G \) and \( O \) and estimate optimal weights from Gradient-Blending. To reduce computation costs, we note that weights estimation can be done on a small subset of data, without perturbing the weights too much (see supplementary materials).

Gradient-Blending algorithms take inputs of training data \( T \), validation set \( V \), \( k \) input modalities \( \{m_i\}_{i=1}^k \) and a joint head \( m_{k+1} \) (Fig. 2c). In practice we can use a subset of training set \( T' \) to measure train loss/ accuracy. To compute the Gradient-Blending weights when training from \( N \) for \( n \) epochs, we provide a Gradient-Blending weight estimation in Algorithm 1. We propose two versions of gradient-blending:

1. **Offline Gradient-Blending** is a simple version of gradient-blending. We compute weights only once, and use a fixed set of weights to train entire epoch. This is very easy to implement. See Algorithm 2.

2. **Online Gradient-Blending** is the full version. We re-compute weights regularly (e.g. every \( n \) epochs – called a super-epoch), and train the model with new weights for a super-epoch. See Algorithm 3.

Empirically, offline performs remarkably well. We compare the two in section 3, with online giving additional gains.

**Algorithm 1: G-B Weight Estimation: GB_Estimate**

**input:** \( \varphi^N \), Model checkpoint at epoch \( N \)

\( n \), # of epochs

**Result:** A set of optimal weights with for \( k + 1 \) losses.

**for** \( i = 1, \ldots, k + 1 \) **do**

- Initialize uni-modal/ naive multi-modal network \( \varphi^N_{m_i} \) from corresponding parameters in \( \varphi^N \);
- Train \( \varphi^N_{m_i} \) for \( n \) epochs on \( T \), resulting model \( \varphi^{N+n}_{m_i} \);
- Compute amount of overfitting \( O^i = O_{N,n} \),
- generalization \( G^i = G_{N,n} \) according to Eq. 3

using \( V \) and \( T' \) for modality \( m_i \);

**end**

Compute a set of loss \( \{w_i^*\}_{i=1}^{k+1} = \frac{1}{Z} \sum_{i=1}^{k+1} \frac{G^i}{O^2} \);

**Algorithm 2: Offline Gradient-Blending**

**input:** \( \varphi^0 \), Initialized model \( N \), # of epochs

**Result:** Trained multi-head model \( \varphi^N \)

Compute per-modality weights \( \{w_i\}_{i=1}^k = GB_Estimate(\varphi^0, N) \);

Train \( \varphi^0 \) with \( \{w_i\}_{i=1}^k \) for \( N \) epochs to get \( \varphi^N \);

**Algorithm 3: Online Gradient-Blending**

**input:** \( \varphi^0 \), Initialized model \( N \), # of epochs

\( n \), super-epoch length

**for** \( i = 0, \ldots, \frac{N}{n} - 1 \) **do**

- Current epoch \( N_i = i \ast n \);
- Compute per-modality weights \( \{w_i\}_{i=1}^k = GB_Estimate(\varphi^{N_i}, N_i + n) \);
- Train \( \varphi^{N_i} \) with \( \{w_i\}_{i=1}^k \) for \( n \) epochs to \( \varphi^{N_i+n} \);

**end**
3. Ablation Experiments

3.1. Experimental setup

Datasets. We use three video datasets for ablations: Kinetics, mini-Sports, and mini-AudioSet. Kinetics is a standard benchmark for action recognition with 260k videos [28] of 400 human action classes. We use the train split (240k) for training and the validation split (20k) for testing. Mini-Sports is a subset of Sports-1M [27], a large-scale classification dataset with 1.1M videos of 487 different fine-grained sports. We uniformly sampled 240k videos from train split and 20k videos from the test split. Mini-AudioSet is a subset of AudioSet [22], a multi-label dataset consisting of 2M videos labeled by 527 acoustic events. AudioSet is very class-unbalanced, so we remove tiny classes and subsample the rest (see supplementary). The balanced mini-AudioSet has 418 classes with 243k videos.

Input preprocessing & augmentation. We consider three modalities: RGB, optical flow and audio. For RGB and flow, we use input clips of 16×224×224 as input. We follow [46] for visual pre-processing and augmentation. For audio, we use log-Mel with 100 temporal frames by 40 Mel filters. Audio and visual are temporally aligned.

Backbone architecture. We use ResNet3D [47] as our visual backbone for RGB and flow and ResNet [25] as our audio model, both with 50 layers. For fusion, we use a two-FC-layer network on concatenated features from visual and audio backbones, followed by one prediction layer.

Training and testing. We train our models with synchronous distributed SGD on GPU clusters using Caffe2 [11], with setup as [47]. We hold out a small portion of training data for weight estimate (8% for Kinetics and mini-Sports, 13% for mini-AudioSet). The final video prediction is made by using center crops of 10 uniformly-sampled clips and averaging the 10 predictions.

3.2. Overfitting Problems in Naive Joint Training

We first compare naive audio-RGB joint training with unimodal audio-only and RGB-only training. Fig. 4 plots the training curves on Kinetics (left) and mini-Sports (right). On both datasets, the audio model overfits the most and video overfits least. The naive joint audio-RGB model has lower training error and higher validation error compared with the video-only model; i.e. naive audio-RGB joint training increases overfitting, explaining the accuracy drop compared to video alone.

We extend the analysis and confirm severe overfitting on other multi-modal problems. We consider all 4 possible combinations of the three modalities (audio, RGB, and optical flow). In every case, the validation accuracy of naive joint training is significantly worse than the best single stream model (Table 1), and training accuracy is almost always higher (see supplementary materials).

| Method               | Clip | V@1 | V@5 |
|----------------------|------|-----|-----|
| Naive Training       | 61.8 | 71.7| 89.6|
| RGB Only             | 63.5 | 72.6| 90.1|
| Offline G-Blend      | 65.9 | 74.7| 91.5|
| Online G-Blend       | 66.9 | 75.8| 91.9|

Table 2: Both offline and online Gradient-Blending outperform Naive late fusion and RGB only. Offline G-Blend is slightly less accurate compared with the online version, but much simpler to implement.

3.3. Gradient-Blending is an effective regularizer

In this ablation, we first compare the performance of online and offline versions of G-Blend. Then we show that G-Blend works with different types of optimizers, including ones with adaptive learning rates. Next, we show G-Blend improves the performance on different multi-modal problems (different combinations of modalities), different model architectures and different tasks.

Online G-Blend Works. We begin with the complete version of G-Blend, online G-Blend. We use an initial super-epoch size of 10 (for warmup), and a super-epoch size of 5 thereafter. On Kinetics with RGB-audio setting, online Gradient-Blending surpasses both uni-modal and naive multi-modal baselines, by 3.2% and 4.1% respectively. The weights for online are in fig. 5a. In general, weights tend to be stable at first with slightly more focused on visual; then we see a transition at epoch 15 where the model does “pre-training” on visual trunk; at epoch 20 A/V trunk got all weights to sync the learning from visual trunk. After that, weights gradually stabilize again with a strong focus on visual learning. We believe that, in general, patterns learned by neural network are different at different stage of training (e.g.[36]), thus the overfitting / generalization behavior also changes during training; this leads to different weights at different stages of the training.

Moreover, we observe that G-Blend always outperforms naive training in the online setting (Fig. 5b). With the same initialization (model snapshots at epoch 0,10,15,....,40), we compare the performance of G-Blend model and naive training after a super-epoch (at epoch 10,15,20,...,45), and G-Blend models always outperform naive training. This shows that G-Blend always provides more generalizable training information, empirically proving proposition 1. Additionally, it shows the relevance of minimizing OGR, as using weights that minimize OGR improves performance of the model. For fair comparison, we fix the main trunk and finetune the classifier for both Naive A/V and G-Blend as we want to evaluate the quality of their backbones. At epoch 25, the gain is small since G-Blend puts almost all weights on A/V head, making it virtually indistinguishable from naive training for that super-epoch.

Offline G-Blend Also Works. Although online G-Blend gives significant gains and addresses overfitting well, it is more complicated to implement, and somewhat slower due
to the extra weight computations. As we will now see, Offline G-Blend can be easily adopted and works remarkably well in practice. On the same audio-RGB setting on Kinetics, offline G-Blend also outperforms uni-modal baseline and naive joint training by a large margin, 2.1% and 3.0% respectively (Table 2), and is only slightly worse than online (-1.1%). Based on such observation, we opt to use offline G-Blend in the rest of the ablations, demonstrating its performance across different scenarios. We speculate the online version will be particularly useful for some cases not covered here, for example a fast-learning low-capacity model (perhaps using some frozen pre-trained features), paired with a high-capacity model trained from scratch.

**Adaptive Optimizers.** Section 2.2 introduced G-Blend in an infinitesimal setting: blending different gradient estimation at a single optimization step and assumes same learning rate for each gradient estimator. This is true for many popular SGD-based algorithms, such as SGD with Momentum. However, the assumption may not be rigorous with adaptive optimization methods that dynamically adjust learning rate for each parameter, such as Adam [32] and AdaGrad [15]. We empirically show that offline Gradient-Blending (Algorithm 2) also works with different optimizers. Since SGD gives the best accuracy among the three optimizers, we opt to use SGD for all of our other experiments.

**Different Modalities.** On Kinetics, we study all combinations of three modalities: RGB, optical flow, and audio.

---

**Figure 4:** Severe overfitting of naive audio-video models on Kinetics and mini-Sports. The learning curves (error-rate) of audio model (A), video model (V), and the naive joint audio-video (AV) model on Kinetics (left) and mini-Sports (right). Solid lines plot validation error while dashed lines show train error. The audio-video model overfits more than visual model and is inferior to the video-only model on validation loss.

**Table 3:** G-Blend on different optimizers. We compare G-Blend with Visual only and Naive AV on two additional optimizers: AdaGrad, and Adam. G-Blend consistently outperforms Visual-Only and Naive AV baselines on all three optimizers.

Table 4 presents comparison of our method with naive joint training and best single stream model. We observe significant gains of G-Blend compared to both baselines on all multi-modal problems. It is worth noting that G-Blend is generic enough to work for more than two modalities.

**Different Architectures.** We conduct experiments on mid-fusion strategy [37], which suffers less overfitting and outperforms visual baseline (Figure 1). On audio-visual setting, Gradient-Blending gives 0.8% improvement (top-1 from 72.8% to 73.6%). On a different fusion architecture with Low-Rank Multi-Modal Fusion (LMF) [35], Gradient-Blending gives 4.2% improvement (top-1 from 69.3% to 73.5%). This suggests Gradient-Blending can be adopted to other fusion strategies besides late-fusion and other fusion architectures besides concatenation.

**Different Tasks/Benchmarks.** We pick the problem of joint audio-RGB model training, and go deeper to compare Gradient-Blending with other regularization methods on different tasks and benchmarks: action recognition (Kinetics), sport classification (mini-Sports), and acoustic event detection (mini-AudioSet). We include three baselines: adding dropout at concatenation layer [43], pre-training single stream backbones then finetuning the fusion model, and blending the supervision signals with equal weights (which is equivalent to naive training with two auxiliary losses). Auxiliary losses are popularly used in multi-task learning, and we extend it as a baseline for multi-modal training.

As presented in Table 5, Gradient-Blending outperforms all baselines by significant margins on both Kinetics and mini-Sports. On mini-AudioSet, G-Blend improves all baselines on mAP, and is slightly worse on mAUC com-
Table 4: Gradient-Blending (G-Blend) works on different multi-modal problems. Comparison between G-Blend with naive late fusion and single best modality on Kinetics. On all 4 combinations of different modalities, G-Blend outperforms both naive late fusion network and best uni-modal network by large margins, and it also works for cases with more than two modalities. G-Blend results are averaged over three runs with different initialization. Variances are small and are provided in supplementary.

| Metric          | RGB + A | RGB + OF | OF + A | RGB + OF + A |
|-----------------|---------|----------|--------|--------------|
| Weights         | [RGB,A,Join]=[0.309,0.495,0.196] | [RGB,OF,Join]=[0.827,0.011,0.162] | [OF,A,Join]=[0.33,0.53,0.01,0.13] | [RGB,OF,A,Join]=[0.33,0.53,0.01,0.13] |
| Clip            | V@1     | V@5      | V@1    | V@5         |
| Uni             | 63.5    | 72.6     | 90.1   | 49.2         |
| Naive           | 61.8    | 71.4     | 89.3   | 46.2         |
| G-Blend         | 65.9    | 74.7     | 91.5   | 54.4         |

Figure 6: Top-Bottom 10 classes based on improvement of G-Blend to RGB model. The improved classes are indeed audio-relevant, while those have performance drop are not very audio semantically-related.

Fig. 6 presents top and bottom 10 classes on Kinetics where G-Blend makes the most and least improvements compared with RGB-only. We observe that improved classes usually have a strong audio-correlation, such as clapping and laughing. For texting, although audio-only has nearly 0 accuracy, when combined with RGB using G-Blend, there are still significant improvements. On bottom-10 classes, we indeed find that audio does not seem to be very semantically relevant (e.g. unloading truck). See supplementary materials for more qualitative analysis.

4. Comparison with State-of-the-Art

In this section, we train our multi-modal networks with deeper backbone architectures using offline Gradient-Blending and compare them with state-of-the-art methods on Kinetics, EPIC-Kitchen [14], and AudioSet. EPIC-Kitchen is a multi-class egocentric dataset with 28K training videos associated with 352 noun and 125 verb classes. For ablations, following [8], we construct a validation set of unseen kitchen environments. G-Blend is trained with RGB and audio input. For Kinetics and EPIC-Kitchen, we use ip-CSN [46] for visual backbone with 32 frames and ResNet for audio backbone, both with 152 layers. For AudioSet, we use R(2+1)D for visual [47] with 16 frames and ResNet for audio, both with 101 layers. We use the same training setup in section 3. For EPIC-Kitchen, we follow the same audio feature extractions as [29]; the visual backbone is pretrained on IG-65M [23]. We use the same evaluation setup as section 3 for AudioSet and EPIC-Kitchen. For Kinetics, we follow the 30-crop evaluation setup as [51]. Our main purposes in these experiments are: 1) to confirm the benefit of Gradient-Blending on high-capacity models; and 2) to compare G-Blend with state-of-the-art methods on different large-scale benchmarks.

Results. Table 6 presents results of G-Blend and compares them with current state-of-the-art methods on Kinetics. First, G-Blend provides an 1.3% improvement over RGB model (the best uni-modal network) with the same backbone architecture ip-CSN-152 [46] when both models are trained from scratch. This confirms that the benefits of G-Blend still hold with high capacity model. Second, G-Blend outperforms state-of-the-arts multi-modal baseline Shift-Attention Network [10] by 1.4% while using less modalities (not using optical flow) and no pre-training. It is on-par with SlowFast [17] while being 2x faster. G-Blend, when fine-tuned from Sports-1M on visual and AudioSet on audio, outperforms SlowFast Network and SlowFast augmented by Non-Local [51] by 1.5% and 0.6% respectively, while being 2x faster than both. Using weakly-supervised pre-training by IG-65M [23] on visual, G-Blend gives unparalleled 83.3% top-1 accuracy and 96.0% top-5 accuracy.

We also note that there are many competitive methods reporting results on Kinetics, due to the space limit, we select only a few representative methods for comparison including Shift-Attention [10], SlowFast [17], and ip-CSN [46]. Shift-Attention and SlowFast are the methods with the best published accuracy using multi-modal and uni-modal input, respectively. ip-CSN is used as the visual backbone of G-
Table 5: G-Blend outperforms all baseline methods on different benchmarks and tasks. Comparison of G-Blend with different regularization baselines as well as uni-modal networks on Kinetics, mini-Sports, and mini-AudioSet. G-Blend consistently outperforms other methods, except for being comparable with using auxiliary loss on mini-AudioSet due to the similarity of learned weights of G-Blend and equal weights.

| Backbone | Pre-train | Validation Set |
|----------|-----------|----------------|
| Shift-Attn Net [10] | ImageNet | Visual: ip-CSN-152 [46] |
| SlowFast [17] | None | Naive A/V: 152 |
| SlowFast+NL [17] | None | TAL-Net [53] |
| ip-CSN-152 [46] | None | G-Blend (ours) |
| G-Blend (ours) | None | Leaderboard [1] |

Table 6: Comparison with state-of-the-art methods on Kinetics. G-Blend used audio and RGB as input modalities; for pre-trained models on Sports1M and IG-65M, G-Blend initializes audio network by pre-training on AudioSet. G-Blend outperforms current state-of-the-art multi-modal method (Shift-Attention Network) despite the fact that it uses fewer modalities (G-Blend does not use Optical Flow). G-Blend also gives a good improvement over RGB model (the best uni-modal network) when using the same backbone, and it achieves the state-of-the-arts.

| Pre-Training | Validation Set |
|--------------|----------------|
| Audio only | Visual:R2D-101 |
| RGB only | Visual:R(2+1)D-101 |
| Pre-Training | Naive A/V:101 |
| Naive | Visual:ip-CSN-152 |
| Dropout | G-Blend (ours) |
| Auxiliary Loss | Leaderboard |
| G-Blend (ours) | Baidu-UTS (leaderboard) |

Table 7: Comparison with state-of-the-art methods on AudioSet. G-Blend outperforms the state-of-the-art methods by a large margin.

| Method | mAP | mAUC |
|--------|-----|------|
| Multi-level Attn. [56] | 0.360 | 0.970 |
| TAL-Net [53] | 0.362 | 0.965 |
| Audio:R2D-101 | 0.324 | 0.961 |
| Visual:R(2+1)D-101 | 0.188 | 0.918 |
| Naive A/V:101 | 0.402 | 0.973 |
| G-Blend (ours):101 | 0.418 | 0.975 |

Table 8: Comparison with state-of-the-art methods on EPIC-Kitchen. G-Blend achieves 2nd place on seen kitchen challenge and 4th place on unseen, despite using fewer modalities, fewer backbones, and single model in contrast to model ensembles compared to published results on leaderboard.

5. Discussion

In uni-modal networks, diagnosing and correcting overfitting typically involves manual inspection of learning curves. Here we have shown that for multi-modal networks it is essential to measure and correct overfitting in a principled way, and we put forth a useful and practical measure of overfitting. Our proposed method, Gradient-Blending, uses this measure to obtain significant improvements over baselines, and either outperforms or is comparable with state-of-the-art methods on multiple tasks and benchmarks. The method potentially applies broadly to end-to-end training of ensemble models, and we look forward to extending G-Blend to other fields where calibrating multiple losses is needed, such as multi-task.
References

[1] Combining correlated unbiased estimators of the mean of a normal distribution. https://projecteuclid.org/download/pdf_1/euclid.lnms/1196285392.

[2] Epic-kitchens action recognition. https://competitions.codalab.org/competitions/20115. Accessed: 2019-11-13.

[3] H. Alamri, V. Cartillier, A. Das, J. Wang, A. Cherian, I. Essa, D. B. amd Tim K. Marks, C. Hori, P. Anderson, S. Lee, and D. Parikh. Audio-visual scene-aware dialog. In CVPR, 2019.

[4] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. L. Zitnick, and D. Parikh. VQA: Visual Question Answering. In ICCV, 2015.

[5] R. Arandjelovi and A. Zisserman. Look, listen and learn. In ICCV, 2017.

[6] J. Arevalo, T. Solorio, M. M. y Gmez, and F. A. Gonzalez. Gated multimodal units for information fusion. In ICLR Workshop, 2017.

[7] T. Baltruvsaitis, C. Ahuja, and L.-P. Morency. Multimodal machine learning: A survey and taxonomy. IEEE Transactions on Pattern Analysis and Machine Intelligence, 41:423–443, 2018.

[8] F. Baradel, N. Neverova, C. Wolf, J. Mille, and G. Mori. Object level visual reasoning in videos. In ECCV, 2018.

[9] R. Bernardi, R. Cakici, D. Elliott, A. Erdem, E. Erdem, N. Ikizler-Cinbis, F. Keller, A. Muscat, and B. Plank. Automatic description generation from images: A survey of models, datasets, and evaluation measures. J. Artif. Int. Res., 55(1):409–442, Jan. 2016.

[10] Y. Bian, C. Gan, X. Liu, F. Li, X. Long, Y. Li, H. Qi, J. Zhou, S. Wen, and Y. Lin. Revisiting the effectiveness of off-the-shelf temporal modeling approaches for large-scale video classification. CoRR, abs/1708.03805, 2017.

[11] Caffe2-Team. Caffe2: A new lightweight, modular, and scalable deep learning framework. https://caffe2.ai/.

[12] J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In CVPR, 2017.

[13] Z. Chen, V. Badrinarayanan, C.-Y. Lee, and A. Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In ICML, 2018.

[14] D. Damen, H. Doughty, G. M. Farinella, S. Fidler, A. Furnari, E. Kazakos, D. Molitsanti, J. Munro, T. Perrett, W. Price, and M. Wray. Scaling egocentric vision: The epic-kitchens dataset. In ECCV, 2018.

[15] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. J. Mach. Learn. Res., 12:2121–2159, July 2011.

[16] D. Eigen and R. Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. ICCV, 2015.

[17] C. Feichtenhofer, H. Fan, J. Malik, and K. He. Slowfast networks for video recognition. In ICCV, 2019.

[18] C. Feichtenhofer, A. Pinz, and R. P. Wildes. Spatiotemporal residual networks for video action recognition. In NIPS, 2016.

[19] C. Feichtenhofer, A. Pinz, and A. Zisserman. Convolutional two-stream network fusion for video action recognition. In CVPR, 2016.

[20] A. Frome, G. S. Corrado, J. Shlens, S. Bengio, J. Dean, M. A. Ranzato, and T. Mikolov. Devise: A deep visual-semantic embedding model. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, NIPS 26, pages 2121–2129. Curran Associates, Inc., 2013.

[21] A. Fukui, D. H. Park, D. Yang, A. Rohrbach, T. Darrell, and M. Rohrbach. Multimodal compact bilinear pooling for visual question answering and visual grounding. In EMNLP, 2016.

[22] J. F. Gemmeke, D. P. W. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter. Audio set: An ontology and human-labeled dataset for audio events. In Proc. IEEE ICASSP 2017, New Orleans, LA, 2017.

[23] D. Ghadiyaram, M. Feiszli, D. Tran, X. Yan, H. Wang, and D. K. Mahajan. Large-scale weakly-supervised pre-training for video action recognition. In CVPR, 2019.

[24] Y. Goyal, T. Khot, D. Summers-Stay, D. Batra, and D. Parikh. Making the V in VQA matter: Elevating the role of image understanding in Visual Question Answering. In CVPR, 2017.

[25] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In CVPR, 2016.

[26] J. Hu, L. Shen, and G. Sun. Squeeze-and-excitation networks. In CVPR, 2018.

[27] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In CVPR, 2014.
[57] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4694–4702, 2015.

[58] P. Zhang, Y. Goyal, D. Summers-Stay, D. Batra, and D. Parikh. Yin and Yang: Balancing and answering binary visual questions. In *CVPR*, 2016.

[59] H. Zhao, C. Gan, A. Rouditchenko, C. Vondrick, J. McDermott, and A. Torralba. The sound of pixels. In *ECCV*, 2018.

---

**A. Estimating Weights on Subsets of Data**

We show that weight estimations by Gradient-Blending is robust on small subsets of data. We sampled 25%, 50% and 75% of Kinetics dataset and use these subsets as train sets in Alg. 2 in main paper. As shown in Fig. 7, the estimated weights are stable on small subsets of data. This suggests that the computational cost of the algorithm can be reduced by using a small subset of data for weight estimation.

---

**B. Understanding OGR**

Overfitting is typically understood as learning patterns in a training set that do not generalize to the target distribution. We quantify this as follows. Given model parameters $\Theta(N)$, where $N$ indicates the training epoch, let $L^T(\Theta(N))$ be the model’s average loss over the fixed training set, and $L^*(\Theta(N))$ be the “true” loss w.r.t the hypothetical target distribution. (In practice, $L^*$ is approximated by the test and validation losses.) For either loss, the quantity $L(\Theta(0)) - L(\Theta(N))$ is a measure of the information gained during training. We define overfitting as the gap between the gain on the training set and the target distribution:

$$O_N \equiv \left( L^T(\Theta(0)) - L^T(\Theta(N)) \right) - \left( L^*(\Theta(0)) - L^*(\Theta(N)) \right)$$

and generalization to be the amount we learn (from training) about the target distribution:

$$G_N \equiv L^*(\Theta(0)) - L^*(\Theta(N))$$

The overfitting-to-generalization ratio is a measure of information quality for the training process of $N$ epochs:

$$OGR = \left| \frac{L^T(\Theta(0)) - L^T(\Theta(N)) - (L^*(\Theta(0)) - L^*(\Theta(N)))}{L^*(\Theta(0)) - L^*(\Theta(N))} \right|$$  (8)

We can also define the amount of overfitting and generalization for an intermediate step from epoch $N$ to epoch

---

**Figure 7: Weight Estimations on Subsets of Data.** We used a small subset of Kinetics dataset to estimate the weights. The weights are quite robust as we decrease the volume of dataset. This suggests feasibility to use subsets to reduce the costs for Gradient-Blending.
N + n, where
\[ \Delta O_{N,n} \equiv (O_{N+n} - O_N) \]
and
\[ \Delta G_{N,n} \equiv (G_{N+n} - G_N) \]
Together, this gives OGR between any two checkpoints:
\[ OGR \equiv \left( \frac{\Delta O_{N,n}}{\Delta G_{N,n}} \right) \]
However, it does not make sense to optimize this as-is. Very underfit models, for example, may still score quite well (difference of train loss and validation loss is very small for underfitting models). What does make sense, however, is to solve an infinitesimal problem: given several estimates of the gradient, blend them to minimize an infinitesimal OGR (or equivalently OGR^2). We can then apply this blend to our optimization process by stochastic gradients (eg. SGD with momentum). In a multi-modal setting, this means we can combine gradient estimates from multiple modalities and minimize OGR to ensure each gradient step now produces a gain no worse than that of the single best modality.

Consider this in an infinitesimal setting (or a single parameter update step). Given parameter Θ, the full-batch gradient with respect to the training set is \( \nabla \mathcal{L}^T(\Theta) \), and the ground truth gradient is \( \nabla \mathcal{L}^*(\Theta) \). We decompose \( \nabla \mathcal{L}^T \) into the true gradient and a remainder:
\[ \nabla \mathcal{L}^T(\Theta) = \nabla \mathcal{L}^*(\Theta) + \epsilon \] (9)
In particular, \( \epsilon = \nabla \mathcal{L}^T(\Theta) - \nabla \mathcal{L}^*(\Theta) \) is exactly the infinitesimal overfitting. Given an estimate \( \hat{g} \) with learning rate \( \eta \), we can measure its contribution to the losses via Taylor’s theorem:
\[ \mathcal{L}^T(\Theta + \eta \hat{g}) \approx \mathcal{L}^T(\Theta) + \eta (\nabla \mathcal{L}^T, \hat{g}) \]
\[ \mathcal{L}^*(\Theta + \eta \hat{g}) \approx \mathcal{L}^*(\Theta) + \eta (\nabla \mathcal{L}^*, \hat{g}) \]
which implies \( \hat{g} \)’s contribution to overfitting is given by \( \langle \nabla \mathcal{L}^T - \nabla \mathcal{L}^*, \hat{g} \rangle \). If we train for \( N \) steps with gradients \( \{\hat{g}_i\}_0^N \), and \( \eta_i \) is the learning rate at \( i \)-th step, the final OGR can be aggregated as:
\[ OGR = \frac{\sum_{i=0}^{N} \eta_i \langle \nabla \mathcal{L}^T(\Theta^{(i)}) - \nabla \mathcal{L}^*(\Theta^{(i)}), \hat{g}_i \rangle}{\sum_{i=0}^{N} \eta_i \langle \nabla \mathcal{L}^*(\Theta^{(i)}), \hat{g}_i \rangle} \] (10)
and OGR^2 for a single vector \( \hat{g}_i \) is
\[ OGR^2 = \left( \frac{\langle \nabla \mathcal{L}^T(\Theta^{(i)}) - \nabla \mathcal{L}^*(\Theta^{(i)}), \hat{g}_i \rangle}{\langle \nabla \mathcal{L}^*(\Theta^{(i)}), \hat{g}_i \rangle} \right)^2 \] (11)
Next we will compute the optimal blend to minimize single-step OGR^2.

C. Proof of Proposition 1

**Proof of Proposition 1.** Without loss of generality, we solve the problem with a different normalization:
\[ \langle \nabla \mathcal{L}^*, \sum_k w_k v_k \rangle = 1 \] (12)
(Note that one can pass between normalizations simply by uniformly rescaling the weights.) With this constraint, the problem simplifies to:
\[ w^* = \arg \min_w \mathbb{E}[\langle \nabla \mathcal{L}^T - \nabla \mathcal{L}^*, \sum_k w_k v_k \rangle^2] \] (13)
We first compute the expectation:
\[ \mathbb{E}[\langle \nabla \mathcal{L}^T - \nabla \mathcal{L}^*, \sum_k w_k v_k \rangle^2] = \mathbb{E}[\langle \sum_k w_k (\nabla \mathcal{L}^T - \nabla \mathcal{L}^*, v_k) \rangle^2] \]
\[ = \mathbb{E}[\sum_{k,j} w_k w_j \langle \nabla \mathcal{L}^T - \nabla \mathcal{L}^*, v_k \rangle \langle \nabla \mathcal{L}^T - \nabla \mathcal{L}^*, v_j \rangle] \]
\[ = \sum_{k,j} w_k w_j \mathbb{E}[\langle \nabla \mathcal{L}^T - \nabla \mathcal{L}^*, v_k \rangle \langle \nabla \mathcal{L}^T - \nabla \mathcal{L}^*, v_j \rangle] \]
\[ = \sum_k w_k^2 \sigma_k^2 \] (14)
where \( \sigma_k^2 = \mathbb{E}[\langle \nabla \mathcal{L}^T - \nabla \mathcal{L}^*, v_k \rangle^2] \) and the cross terms vanish by assumption.
We apply Lagrange multipliers on our objective function (14) and constraint (12):
\[ L = \sum_k w_k^2 \sigma_k^2 - \lambda \left( \sum_k w_k \langle \nabla \mathcal{L}^*, v_k \rangle - 1 \right) \] (15)
The partials with respect to \( w_k \) are given by
\[ \frac{\partial L}{\partial w_k} = 2 w_k \sigma_k^2 - \lambda \langle \nabla \mathcal{L}^*, v_k \rangle \] (16)
Setting the partials to zero, we obtain the weights:
\[ w_k = \lambda \frac{\langle \nabla \mathcal{L}^*, v_k \rangle}{2 \sigma_k^2} \] (17)
The only remaining task is obtaining the normalizing constant. Applying the constraint gives:
\[ 1 = \sum_k w_k \langle \nabla \mathcal{L}^*, v_k \rangle = \lambda \sum_k \frac{\langle \nabla \mathcal{L}^*, v_k \rangle^2}{2 \sigma_k^2} \] (18)
In other words,
\[ \lambda = \frac{2}{\sum_k \frac{\langle \nabla \mathcal{L}^*, v_k \rangle^2}{\sigma_k^2}} \] (19)
Setting $Z = 1/\lambda$ we obtain $w_k^* = \frac{1}{Z} \frac{(\sum L^T, v_k)^2}{\lambda \sigma^2}$. Dividing by the sum of the weights yields the original normalization. \hfill \Box

Note: if we relax the assumption that $E[(\nabla L^T - \nabla L^*, v_k)(\nabla L^T - \nabla L^*, v_j)] = 0$ for $k \neq j$, the proof proceeds similarly, although from (14) it becomes more convenient to proceed in matrix notation. Define a matrix $\Sigma$ with entries given by

$$
\Sigma_{kj} = E[(\nabla L^T - \nabla L^*, v_k)(\nabla L^T - \nabla L^*, v_j)]
$$

Then one finds that

$$
w_k^* = \frac{1}{Z} \sum_j \Sigma_{kj}^{-1} \langle \nabla L^*, v_k \rangle
$$

$$
Z = \frac{1}{2} \sum_{k,j} \Sigma_{kj}^{-1} \langle \nabla L^*, v_k \rangle^2
$$

D. Variances of G-Blend Runs

The variances of the performances on the datasets used by the paper are typically small, and previous works provide results on a single run. To verify that G-Blend results are reproducible, we conducted multiple runs for G-Blend results in Table 3 of the main paper. We found that the variance is consistent across different modalities for G-Blend results (Table 9).

E. Sub-sampling and Balancing Multi-label Dataset

For a single-label dataset, one can subsample and balance at a per-class level such that each class may have the same volume of data. Unlike single-label dataset, classes in multi-label dataset can be correlated. As a result, sampling a single data may add volume for more than one class. This makes the naive per-class subsampling approach difficult.

To uniformly sub-sample and balance AudioSet to get mini-AudioSet, we propose the following algorithm:

**Algorithm 4:** Sub-sampling and Balancing Multi-label Dataset

**Data:** Original Multi-Class Dataset $\mathcal{D}$, Minimum Class Threshold $M$, Target Class Volume $N$

**Result:** Balanced Sub-sampled Multi-label Dataset $\mathcal{D}'$

Initialize empty dataset $\mathcal{D}'$;

Remove labels from $\mathcal{D}$ such that label volume is less than $M$;

Randomly shuffle entries in $\mathcal{D}$;

for Data Entry $d \in \mathcal{D}$ do

Choose class $c$ of $d$ such that the volume of $c$ is the smallest in $\mathcal{D}'$;

Let the volume of $c$ be $V_c$ in $\mathcal{D}$;

Let the volume of $c$ be $V'_c$ in $\mathcal{D}'$;

Generate random number $r$ to be an integer between 0 and $V_c - V'_c$;

if $r < N - V'_c$ then

| Select $d$ to $\mathcal{D}'$;

else

| Skip $d$ and continue;

end

We noticed that the results of all these approaches are sub-optimal. We speculate that less layers may fail to fully learn the relations of the features, while deeper fusion network overfits more.

F. Details on Model Architectures

F.1. Late Fusion By Concatenation

In late fusion by concatenation strategy, we concatenate the output features from each individual network (i.e. $k$ modalities’ 1-D vectors with $n$ dimensions). If needed, we add dropout after the feature concatenations.

The fusion network is composed of two $FC$ layers, with each followed by an $ReLU$ layer, and a linear classifier. The first $FC$ maps $kn$ dimensions to $n$ dimensions, and the second one maps $n$ to $n$. The classifier maps $n$ to $c$, where $c$ is the number of classes.

As sanity check, we experimented using less or more $FC$ layers on Kinetics:

- **0 FC.** We only add a classifier that maps $kn$ dimensions to $c$ dimensions.
- **1 FC.** We add one $FC$ layer that maps $kn$ dimensions to $n$ dimension, followed by an $ReLU$ layer and classifier to map $n$ dimension to $c$ dimensions.
- **4 FC.** We add one $FC$ layer that maps $kn$ dimensions to $n$ dimension, followed by an $ReLU$ layer. Then we add 3 $FC$-$ReLU$ pairs that preserve the dimensions. Then we add an a classifier to map $n$ dimension to $c$ dimensions.

We noticed that the results of all these approaches are sub-optimal. We speculate that less layers may fail to fully learn the relations of the features, while deeper fusion network overfits more.

F.2. Mid Fusion By concatenation

Inspired by [37], we also concatenate the features from each stream at an early stage rather than late fusion. The problem with mid fusion is that features from individual streams can have different dimensions. For example, audio features are 2-D (time-frequency) while visual features are 3-D (time-height-width).

We propose three ways to match the dimension, depending on the output dimension of the concatenated features:

- **1-D Concat.** We downsample the audio features to 1-D by average pooling on the frequency dimension. We downsample the visual features to 1-D by average pooling over the two spatial dimensions.
Table 9: Last row of Table 3 in main papers with variance. Results are averaged over three runs with random initialization, and ± indicates variances.

|               | RGB + A | RGB + OF | OF + A | RGB + OF + A |
|---------------|---------|----------|--------|--------------|
| Clip V@1     | 65.9 ± 0.1 | 91.5 ± 0.1 | 54.4 ± 0.6 | 66.1 ± 0.4 |
| V@5          | 74.7 ± 0.2 | 64.3 ± 0.1 | 66.3 ± 0.5 | 74.9 ± 0.2 |

- **2-D Concat.** We keep the audio features the same and match the visual features to audio features. We downsample the visual features to 1-D by average pooling over the two spatial dimensions. Then we tile the 1-D visual features on frequency dimension to make 2-D visual features.
- **3-D Concat.** We keep the visual features fixed and match the audio features to visual features. We downsample the audio features to 1-D by average pooling over the frequency dimension. Then we tile the 1-D visual features on two spatial dimensions to make 3-D features.

The temporal dimension may also be mismatched between the streams: audio stream is usually longer than visual streams. We add convolution layers with stride of 2 to downsample audio stream if we are performing 2-D concat. Otherwise, we upsample visual stream by replicating features on the temporal dimension.

There are five blocks in the backbones of our ablation experiments (section 4), and we fuse the features using all three strategies after block 2, block 3, and block 4. Due to memory issue, fusion using 3-D concat after block 2 is unfeasible. On Kinetics, we found 3-D concat after block 3 works the best, and it’s reported in Fig. 1 in the main paper. In addition, we found 2-D concat works the best on AudioSet and uses less GFLOPs than 3-D concat. We speculate that the method for dimension matching is task-dependent.

**F.3. SE Gate**

Squeeze-and-Excitement network introduced in [26] applies a self-gating mechanism to produce a collection of per-channel weights. Similar strategies can be applied in a multi-modal network to take inputs from one stream and produce channel weights for the other stream.

Specifically, we perform global average pooling on one stream and use the same architectures in [26] to produce a set of weights for the other channel. Then we scale the channels of the other stream using the weights learned. We either do a ResNet-style skip connection to add the new features or directly replace the features with the scaled features. The gate can be applied from one direction to another, or on both directions. The gate can also be added at different levels for multiple times. We found that on Kinetics, it works the best when applied after block 3 and on both directions.

We note that we can also first concatenate the features and use features from both streams to learn the per-channel weights. The results are similar to learning the weights with a single stream.

**F.4. NL Gate**

Although lightweight, SE-gate fails to offer any spatial-temporal or frequency-temporal level attention. One alternative way is to apply an attention-based gate. We are inspired by the Query-Key-Value formulation of gates in [48]. For example, if we are gating from audio stream to visual stream, then visual stream is Query and audio stream is Key and Value. The output has the same spatial-temporal dimension as Query.

Specifically, we use Non-Local gate in [51] as the implementation for Query-Key-Value attention mechanism. Details of the design are illustrated in fig. 8. Similar to SE-gate, NL-Gate can be added with multiple directions and at multiple positions. We found that it works the best when added after block 4, with a 2-D concat of audio and RGB features as Key-Value and visual features as Query to gate the visual stream.
| Dataset   | Modality     | Validation V@1 | Train V@1 |
|-----------|--------------|----------------|-----------|
| Kinetics  | A            | 19.7           | 85.9      |
|           | RGB          | 72.6           | 90.0      |
|           | OF           | 62.1           | 75.1      |
|           | A + RGB      | 71.4           | 95.6      |
|           | RGB + OF     | 71.3           | 91.9      |
|           | A + OF       | 58.3           | 83.2      |
|           | A + RGB + OF | 70.0           | 96.5      |
| mini-Sport| A            | 22.1           | 56.1      |
|           | RGB          | 62.7           | 77.6      |
|           | A + RGB      | 60.2           | 84.2      |

Table 10: **Multi-modal networks have lower validation accuracy but higher train accuracy.** Table of Top-1 accuracy of single stream models and naive late fusion models. Single stream modalities include RGB, Optical Flow (OF), and Audio Signal (A). Its higher train accuracy and lower validation accuracy signal severe overfitting.

G. Additional Ablation Results

G.1. A strong oracle baseline

In section 3.3, we presented the results on Gradient-Blending as an effective regularizer to train multi-modal networks. Here, we consider an additional strong baseline for the Kinetics, audio-RGB case.

Suppose we have an oracle to choose the best modality (from audio, RGB and naive A/V) for each class. For example, for “whistling” video, the oracle chooses naive A/V model as it performs the best among the three on “whistling” in validation set. With this oracle, Top-1 video accuracy is 74.1%, or 0.6% lower than the offline G-Blend result.

G.2. Training Accuracy

In section 3.2, we introduced the overfitting problem of joint training of multi-modal networks. Here we include both validation accuracy and train accuracy of the multi-modal problems (Table 10). We demonstrate that in all cases, the multi-modal networks are performing worse than their single best counterparts, while almost all of their train accuracy are higher (with the sole exception of OF+A, whose train accuracy is similar to audio network’s train accuracy).

G.3. Early Stopping

In early stopping, we experimented with three different stopping schedules: using 25%, 50% and 75% of iterations per epoch. We found that although overfitting becomes less of a problem, the model tends to under-fit. In practice, we still found that the 75% iterations scheduling works the best among the three, though it’s performance is worse than full training schedule that suffers from overfitting. We summarize their learning curves in fig. 9.

Figure 9: **Early stopping avoids overfitting but tends to under-fit.** Learning curves for three early stopping schedules we experiment. When we train the model with less number of iterations, the model does not overfit, but the undesirable performance indicates an under-fitting problem instead.

G.4. Additional Qualitative Analysis

In section 3.3 we presented the qualitative analysis of G-Blend’s performance compared with RGB model performance (fig.6). We expand the analysis and provide more details in this section.

We first expand the analysis to compare the top-20 and bottom-20 improved classes of G-Blend versus RGB model (fig. 10). This is a direct extension of fig.6. It further confirms that classes that dropped are indeed not very semantically relevant in audio, and in many of those classes, the audio model’s performance is almost 0.

We further extends the analysis to compare naively trained audio-visual model with RGB-only model (fig. 11). We note that the improvement for top-20 classes is smaller than that of G-B and for bot-20 classes the drop is more significant. Moreover, we note that in some bot-20 classes like snorkeling or feeding bird, where the sound of breathing and birds is indeed relevant, naively trained A/V model is not performing well. For these classes, audio model achieves decent performance. We further note that interestingly, for laughing, although naive A/V model outperforms RGB model, it is worse than audio-only model. And only with G-Blend, it benefits from both visual and audio signals, performing better than both.

Finally, we compare the top-20 and bot-20 classes where G-Blend has the most improvement/ drop with naively trained A/V model. We note that the gains in improved classes are much larger than the decrease in dropped classes.
Figure 10: **Top-Bottom 20 classes based on improvement of G-Blend to RGB model.** The improved classes are indeed audio-relevant, while those have performance drop are not very audio semantically-related.

Figure 11: **Top-Bottom 20 classes based on improvement of naively trained audio-visual to RGB model.** The improvement tends to be smaller than that of G-B counterpart and the drop is more significant. More interesting, in some classes, the naively trained A/V model performs worse than audio signal.

Figure 12: **Top-Bottom 20 classes based on improvement of G-Blend to Naive audio-visual model.** We note that the gain is much more significant than drop.