What makes training multi-modal classification networks hard? Supplementary material

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

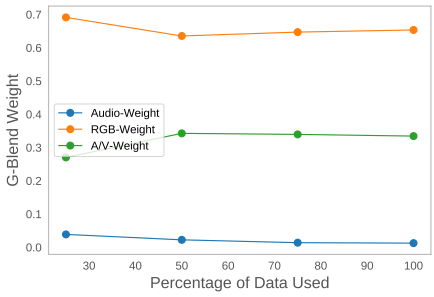

Figure 1: **Weight Estimations on Subsets of Data**. We used a small subset of Kinetics dataset to estimate the weights. The weights are quite robust as we decrease the volume of dataset. This suggests feasibility to use subsets to reduce the costs for Gradient-Blending.

1. 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. 1, 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.

2. 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 $\mathcal{L}^{\mathcal{T}}(\Theta^{(N)})$ be the model's average loss over the fixed training set, and $\mathcal{L}^{*}(\Theta^{(N)})$ be the "true" loss w.r.t the hypothetical target distribution. (In practice, $\mathcal{L}^{*}$ is approximated by the test and validation losses.) For either loss, the quantity $\mathcal{L}(\Theta^{(0)}) - \mathcal{L}(\Theta^{(N)})$ 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:

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

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

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

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

$$OGR = \left| \frac{\left(\mathcal{L}^{\mathcal{T}}(\Theta^{(0)}) - \mathcal{L}^{\mathcal{T}}(\Theta^{(N)})\right) - \left(\mathcal{L}^{*}(\Theta^{(0)}) - \mathcal{L}^{*}(\Theta^{(N)})\right)}{\mathcal{L}^{*}(\Theta^{(0)}) - \mathcal{L}^{*}(\Theta^{(N)})} \right| \quad (1)$$

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

$$OGR \equiv \langle \frac{\Delta O_{N,n}}{\Delta G_{N,n}} \rangle$$

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 $\nabla\mathcal{L}^{\mathcal{T}}(\Theta)$, and the

groundtruth gradient is $\nabla\mathcal{L}^*(\Theta)$. We decompose $\nabla\mathcal{L}^{\mathcal{T}}$ into the true gradient and a remainder:

$$\nabla\mathcal{L}^{\mathcal{T}}(\Theta) = \nabla\mathcal{L}^*(\Theta) + \epsilon \tag{2}$$

In particular, $\epsilon = \nabla\mathcal{L}^{\mathcal{T}}(\Theta) - \nabla\mathcal{L}^*(\Theta)$ is exactly the infinitesimal overfitting. Given an estimate $\hat{g}$ with learning rate η, we can measure its contribution to the losses via Taylor's theorem:

$$\mathcal{L}^{\mathcal{T}}(\Theta + \eta\hat{g}) \approx \mathcal{L}^{\mathcal{T}}(\Theta) + \eta\langle\nabla\mathcal{L}^{\mathcal{T}}, \hat{g}\rangle$$
$$\mathcal{L}^*(\Theta + \eta\hat{g}) \approx \mathcal{L}^*(\Theta) + \eta\langle\nabla\mathcal{L}^*, \hat{g}\rangle$$

which implies $\hat{g}$'s contribution to overfitting is given by $\langle\nabla\mathcal{L}^{\mathcal{T}} - \nabla\mathcal{L}^*, \hat{g}\rangle$. If we train for N steps with gradients $\{\hat{g}_i\}_0^N$, and η_i is the learning rate at i-th step, the final OGR can be aggregated as:

$$OGR = \left| \frac{\sum_{i=0}^{N} \eta_i \langle\nabla\mathcal{L}^{\mathcal{T}}(\Theta^{(i)}) - \nabla\mathcal{L}^*(\Theta^{(i)}), \hat{g}_i\rangle}{\sum_{i=0}^{N} \eta_i \langle\nabla\mathcal{L}^*(\Theta^{(ni)}), \hat{g}_i\rangle} \right| \tag{3}$$

and OGR^2 for a single vector $\hat{g}_i$ is

$$OGR^2 = \left(\frac{\langle\nabla\mathcal{L}^{\mathcal{T}}(\Theta^{(i)}) - \nabla\mathcal{L}^*(\Theta^{(i)}), \hat{g}_i\rangle}{\langle\nabla\mathcal{L}^*(\Theta^{(i)}), \hat{g}_i\rangle} \right)^2 \tag{4}$$

Next we will compute the optimal blend to minimize single-step OGR^2.

3. Proof of Proposition 1

Proof of Proposition 1. Without loss of generality, we solve the problem with a different normalization:

$$\langle\nabla\mathcal{L}^*, \sum_k w_k v_k\rangle = 1 \tag{5}$$

(Note that one can pass between normalizations simply by uniformly rescaling the weights.) With this constraint, the problem simplifies to:

$$w^* = \arg\min_w \mathbb{E}[(\langle\nabla\mathcal{L}^{\mathcal{T}} - \nabla\mathcal{L}^*, \sum_k w_k v_k\rangle)^2] \tag{6}$$

We first compute the expectation:

$$\begin{aligned}
&\mathbb{E}[(\langle\nabla\mathcal{L}^{\mathcal{T}} - \nabla\mathcal{L}^*, \sum_k w_k v_k\rangle)^2] \\
&= \mathbb{E}[(\sum_k w_k \langle\nabla\mathcal{L}^{\mathcal{T}} - \nabla\mathcal{L}^*, v_k\rangle)^2] \\
&= \mathbb{E}[\sum_{k,j} w_k w_j \langle\nabla\mathcal{L}^{\mathcal{T}} - \nabla\mathcal{L}^*, v_k\rangle\langle\nabla\mathcal{L}^{\mathcal{T}} - \nabla\mathcal{L}^*, v_j\rangle] \\
&= \sum_{k,j} w_k w_j \, \mathbb{E}\left[\langle\nabla\mathcal{L}^{\mathcal{T}} - \nabla\mathcal{L}^*, v_k\rangle\langle\nabla\mathcal{L}^{\mathcal{T}} - \nabla\mathcal{L}^*, v_j\rangle\right] \\
&= \sum_k w_k^2 \sigma_k^2
\end{aligned} \tag{7}$$

where $\sigma_k^2 = \mathbb{E}[\langle\nabla\mathcal{L}^{\mathcal{T}} - \nabla\mathcal{L}^*, v_k\rangle^2]$ and the cross terms vanish by assumption.

We apply Lagrange multipliers on our objective function (7) and constraint (5):

$$L = \sum_k w_k^2 \sigma_k^2 - \lambda \left(\sum_k w_k \langle\nabla\mathcal{L}^*, v_k\rangle - 1 \right) \tag{8}$$

The partials with respect to w_k are given by

$$\frac{\partial L}{\partial w_k} = 2 w_k \sigma_k^2 - \lambda\langle\nabla\mathcal{L}^*, v_k\rangle \tag{9}$$

Setting the partials to zero, we obtain the weights:

$$w_k = \lambda \frac{\langle\nabla\mathcal{L}^*, v_k\rangle}{2\sigma_k^2} \tag{10}$$

The only remaining task is obtaining the normalizing constant. Applying the constraint gives:

$$1 = \sum_k w_k \langle\nabla\mathcal{L}^*, v_k\rangle = \lambda \sum_k \frac{\langle\nabla\mathcal{L}^*, v_k\rangle^2}{2\sigma_k^2} \tag{11}$$

In other words,

$$\lambda = \frac{2}{\sum_k \frac{\langle\nabla\mathcal{L}^*, v_k\rangle^2}{\sigma_k^2}} \tag{12}$$

Setting $Z = 1/\lambda$ we obtain $w_k^* = \frac{1}{Z} \frac{\langle\nabla\mathcal{L}^*, v_k\rangle^2}{2\sigma_k^2}$. Dividing by the sum of the weights yields the original normalization. □

Note: if we relax the assumption that $\mathbb{E}[\langle\nabla\mathcal{L}^{\mathcal{T}} - \nabla\mathcal{L}^*, v_k\rangle\langle\nabla\mathcal{L}^{\mathcal{T}} - \nabla\mathcal{L}^*, v_j\rangle] = 0$ for $k \neq j$, the proof proceeds similarly, although from (7) it becomes more convenient to proceed in matrix notation. Define a matrix Σ with entries given by

$$\Sigma_{kj} = \mathbb{E}[\langle\nabla\mathcal{L}^{\mathcal{T}} - \nabla\mathcal{L}^*, v_k\rangle\langle\nabla\mathcal{L}^{\mathcal{T}} - \nabla\mathcal{L}^*, v_j\rangle]$$

Then one finds that

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

4. 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 1).

RGB + A			RGB + OF			OF + A			RGB + OF + A		
Clip	V@1	V@5	Clip	V@1	V@5	Clip	V@1	V@5	Clip	V@1	V@5
65.9±0.1	**74.7±0.2**	**91.5±0.1**	**64.3±0.1**	**73.1±0.0**	**90.8±0.1**	**54.4±0.6**	**66.3±0.5**	**86.0±0.6**	**66.1±0.4**	**74.9±0.2**	**91.8±0.2**

Table 1: **Last row of Table 3 in main papers with variance.** Results are averaged over three runs with random initialization, and ± indicates variances.

5. 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 1: 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
end

6. Details on Model Architectures

6.1. 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.

6.2. Mid Fusion By concatenation

Inspired by [2], 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.
- **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.

6.3. SE Gate

Squeeze-and-Excitement network introduced in [1] 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 [1] 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.

6.4. 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 [3]. 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 [4] as the implementation for Query-Key-Value attention mechanism. Details of the design are illustrated in fig. 2. 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.

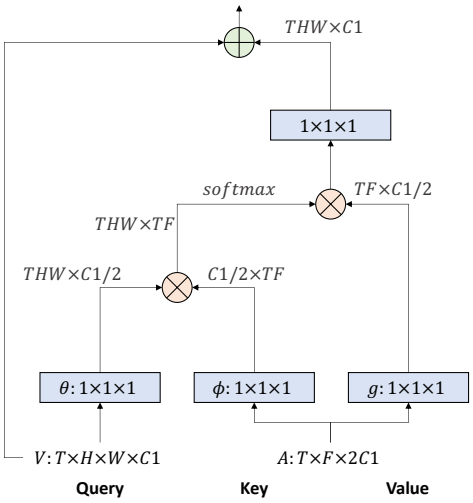

Figure 2: **NL-Gate Implementation**. Figure of the implementation of NL-Gate on visual stream. Visual features are the Query. The 2D Mid-Concatenation of visual and audio features is the Key and Value.

7. Additional Ablation Results

7.1. 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.

7.2. 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 2). 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).

7.3. 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

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 2: **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.

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. 3.

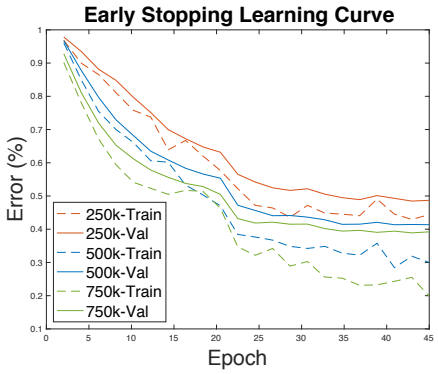

Figure 3: **Early stopping avoids overfitting but tends to under-fit.** Learning curves for three early stopping schedules we experiment. When we train the model with less number of iterations, the model does not overfit, but the undesirable performance indicates an under-fitting problem instead.

7.4. 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. 4). 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. 5). We note that the improvement for top-20 classes is smaller than that of G-B and for bot-20 classes the drop is mroe 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.

References

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

[2] A. Owens and A. A. Efros. Audio-visual scene analysis with self-supervised multisensory features. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. 2017.

[4] X. Wang, R. Girshick, A. Gupta, and K. He. Non-local neural networks. In *CVPR*, 2018.

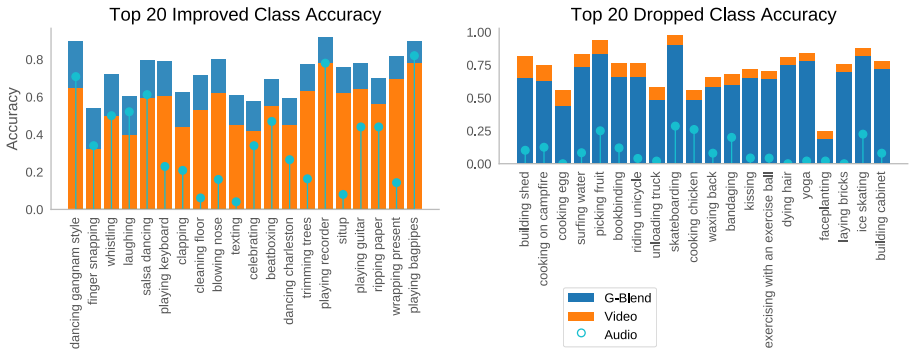

Figure 4: **Top-Bottom 20 classes based on improvement of G-Blend to RGB model.** The improved classes are indeed audio-relevant, while those have performance drop are not very audio semantically-related.

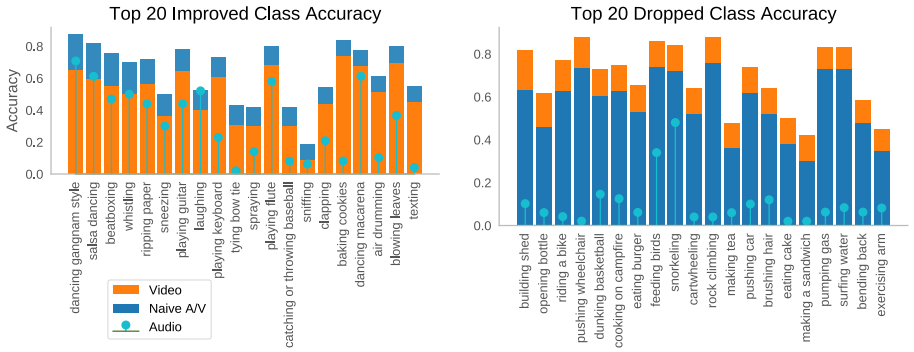

Figure 5: **Top-Bottom 20 classes based on improvement of naively trained audio-visual to RGB model.** The improvement tends to be smaller than that of G-B counterpart and the drop is more significant. More interesting, in some classes, the naively trained A/V model performs worse than audio signal.

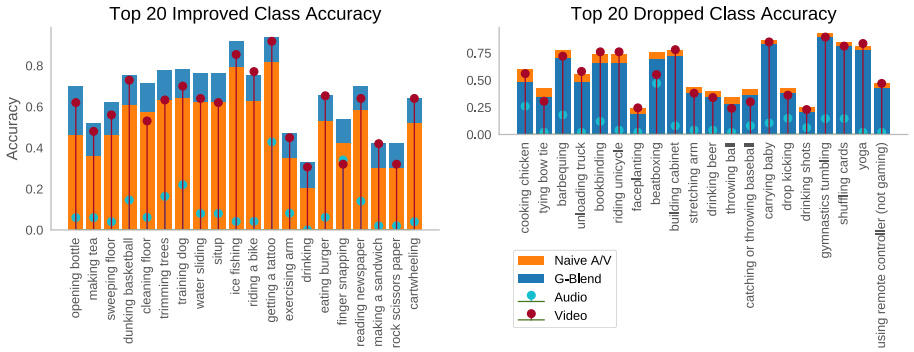

Figure 6: **Top-Bottom 20 classes based on improvement of G-Blend to Naive audio-visual model.** We note that the gain is much more significant than drop.