Improving Webly Supervised Learning

Noppameth Noeimuangpak
Carnegie Mellon University
Pittsburgh, PA 15213
nnoeimua@andrew.cmu.edu

Klaijan Sinteppadon
Carnegie Mellon University
Pittsburgh, PA 15213
ksintepp@andrew.cmu.edu

Siwapol Techaratsami
Carnegie Mellon University
Pittsburgh, PA 15213
stechara@andrew.cmu.edu

Khanin Udomchoksakul
Carnegie Mellon University
Pittsburgh, PA 15213
kudomcho@andrew.cmu.edu

Bhiksha Raj
Language Technologies Institute,
School of Computer Science,
Carnegie Mellon University
Pittsburgh, PA 15213
bhikshar@cs.cmu.edu

Ankit Shah
Language Technologies Institute,
School of Computer Science,
Carnegie Mellon University
Pittsburgh, PA 15213
aps1@cs.cmu.edu

Abstract

Co-teaching method on weakly labeled sound data has gained increasing attention due to its capability to help classify unlabeled data usually done by human labelers. Our recent application of such technique was applied to the sound events domain, with attempts from an academic branch to achieve this through an online dataset. WeblyNet[11], developed by the Language Institute Technologies, shows a promising learning technique that learns sound from webly labeled data, demonstrating the mechanism by introducing two networks co-teaching each other. One potential consideration of this technique is that no verification of network's learning independence exists; thus no such guarantee that two or more networks are in fact learning different aspects of data. Our solution is to add an additional term on the agreement step to penalize the overall loss to reduce the learning dependence among networks. We experiment with two approaches: modifying the KL divergence with masked KL, and applying similarity penalty with cosine similarity and centered kernel alignment onto the agreement loss component. Our result shows that models perform better and converge faster than the baseline when incorporating our implementation in specific scenarios.

1 Introduction

Classifying weakly labeled sound data has been a challenge due to the absent of human manual labeling involved in the process. The weakly labeled dataset, usually retrieved online and labeled without verification, pose difficulties in enabling the network to learn correctly. Several experiments have attempted to resolve the training process from image, yet it is not prevalent as of recent in the research involving sound data.

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

Kumar, et al.[11] proposed WeblyNet that learns from weakly labeled data by introducing system with two models co-teaching each other from different views of data, one that takes the embedding layers from the pre-trained model and the other that takes the raw input data. To measure classification agreement between two models, they created an agreement loss that contains both models' individual loss, and the generalized KL divergence between outputs from 2 networks. These three terms are combined in the overall loss, which will be minimized when training. This approach shows promising results for learning from weakly labelled data compared to training with only one network.

One aspect regarding WeblyNet system and weakly labelled data, webly data, that has not been investigated more extensive is measuring independence among the models, which would help ensure that two models are not learning the same thing, or one model is not too influential to the other. This work assumes independence only at different views of data but not in learning process among classifiers during predicting these noisy labeled data.

Our goal is to create mechanism(s) that help increases networks learning independence while not compromising classification performance overall. We propose two approaches to enhance independence, which are 1) apply similarity penalty to the overall loss, and 2) modify the existing overall loss with KL masking loss to reduce dependencies among networks.

The following sections of the paper are a literature review, discuss methodologies used in our experiments, evaluate results, conclude our work, and discuss further implementations that could be considered in the next iteration.

2 Literature review

2.1 Learning Sound Events from Webly Labeled Data[11]

Research has been conducted on improving learning approaches of weakly labeled data. Kumar, et al.[11] proposed the WeblyNet that learns by having two models to co-teach each other with weakly labeled data, which update the models through combination of individual losses and divergence measure between outputs of the networks as the consensus mechanism. They trained with webly labeled data retrieved from the internet, and each model is given with a different view of the data when training.

The proposed architecture of the network contains two different networks. The first network (N1) is trained on the first representation of webly labeled data which is embedded in 128-dimensional quantized vectors for each 1 second of audio. The second network (N2) is trained on the second representation of webly labeled data which obtained through transfer learning with AudioSet trained on N1. During the training of the combined network, they used a loss function that combined losses of the two networks individually with respect to the target, and a divergence measure between the outputs of the two networks. The result showed that the improvement was approximately at 17% relative to the baseline.

One consideration is that although they are trained with different features of the data, their divergence term fails to take independence learning into account. Their combined losses are calculated together without addressing loss independence measurement. Our work aims to extend such measurement, and implement an additional mechanism to minimize learning dependence of networks.

2.2 Large Scale Audiovisual Learning of Sounds with Weakly Labeled Data[2]

Multi-modal learning approach with weakly labeld data has shown a promising improvement in enabling better predictions such as classifying sounds using audio and visual models. Haytham M., et al. constructed an attention model that combines two predictions from audio and visual models and generates learnable parameters using an attention model to perform audiovisual classification from the AudioSet dataset. By using Multiple Instance Learning (MIL) and model fusion approaches mentioned above, they achieved a Mean Average Precision (MAP) of 46.16, which is higher than prior state-of-the-art models. Their work shows that multi-modal learning is suitable to perform classification when only weakly labeled data is available.

2.3 SVCCA: Singular Vector Canonical Correlation Analysis for Deep Learning Dynamics and Interpretability [14]

Raghu, et al. showed that Singular Vector Canonical Correlation Analysis can be used to compare representations between two models through layers. This allows measuring similarity of intrinsic dimensionality when learning. Comparing to another similarity measurements like cosine similarity, SVCCA can be used to compare two representations in a way that is both invariant to affine transform. They apply SVCCA to gain insights regarding the learning process motivating new training algorithms that simultaneously save computation and overfit less. Inspired by their study, having a similarity index measuring correlation between model representations can be helpful to penalize the WeblyNet[11] when sub-models are too similar.

2.4 Insights on representational similarity in neural networks with canonical correlation [13]

Morcos, et al. [13] demonstrated using projection weighted CCA to compare different model representations on the convolutional network and recurrent neural network, in order to observe representations change over time during training process. We wish to adopt the idea and implement a similar approach in our implementation during the training of two networks, and penalize the overall loss if their representations are similar.

2.5 Similarity of Neural Network Representations Revisited[8]

Another approach that helps measuring similarity when having different architectures of models is centered kernel alignment (CKA). Simon, et al.[8] presented CKA which is closely related to canonical correlation analysis (CCA). But unlike CCA, CKA can determine correspondences within the hidden layers in networks trained from different model initialization. Several works used this method to identify similarity among image classifiers and we take this opportunity to experiment with CKA when we penalize the overall loss at training.

3 Model description

In our approach, we adopt WeblyNet system architecture proposed by Kumar, et al.[11] as our baseline model, which achieved high performance on webly data. However, this approach might fail to exclude some degrees of dependence among networks. We would like to investigate explicit approaches that can force independence in the system to improve the model performance.

Our study contributes in the following points:

1. Re-implement our baseline model WeblyNet system[11].
2. Implement WeblyNet system framework with additional similarity loss function added to the total loss function to induce independence between sub-networks. We experiment on candidates for additional loss function such as cosine similarity and centered kernel alignment (CKA).
3. Modify networks' outputs divergence term used by the baseline so that it gives different weights to positive and negative instance. We made hypothesis that this modification would prevent different models to generate the same outputs.

3.1 Baseline model: WeblyNet system

We re-implement the baseline model based on original code in GitHub[11]. In the proposed WeblyNet model, two different networks operate on different views of the data to prevent models from learning the same thing which lead to two models turned to be identical. During the inference, the predictions from two networks are combined so that they can improve the performance of the whole system. During the training, they use a combined loss taken from the individual loss of each model, and the additional term which measures the agreement between two networks.

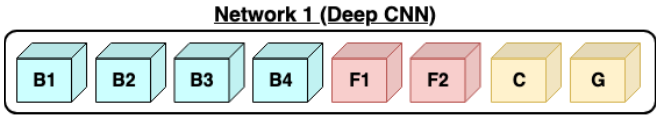


Figure 1: Network 1 architecture retrieved from [11]

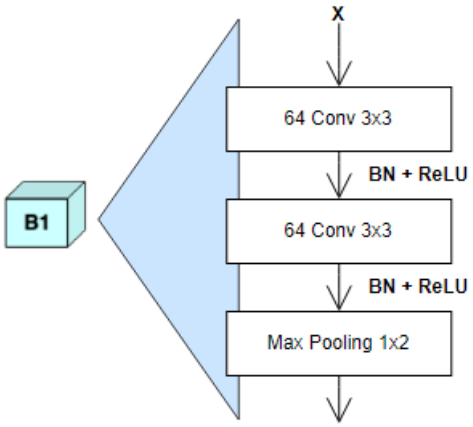


Figure 2: Block of layers in Network 1 retrieved from [11]

3.1.1 Network 1: Deep CNN

Block of layers Each block contains two convolutional layers with a kernel size of 3×3, followed by one max pooling layer with a kernel size of 1×2. Each block also contains batch normalization and ReLU activation after every layers. All convolutional layers operate with a stride of 1 with padding of 1 applied to their inputs. The number of filters used in these blocks are B1:64, B2:128, B3:256, B4:256.

Convolutional layers Convolutional layer F1 has 1,024 filters of size 1×8. Convolutional layer F2 has 1024 filters of size 1×1. These layers are also followed by batch normalization and ReLU activation. These layers operate with a stride of 1 with padding of 1 applied to their inputs.

Segment level output layer (C) This layer is a convolutional layer with 1×1 filters. The number of filters is the same as the number of classes in the dataset (C). This layer is followed by the sigmoid activation function.

Global average pooling (G) The outputs from the segment level are pooled using the global average pooling layer. As a result, this layer produces class outputs with size 1×1.

3.1.2 Network 2: Fully Connected Linear Network

Network 2 is fully connected layers with hidden layers size of 2048, 1024, 1024 followed by an output layer with C number of neurons. After the first and second hidden layers, there are dropouts with a probability of 0.4. All hidden layers use ReLU activation, while the out layer uses sigmoid activation. Figure 3 illustrates the Network 2 layers.

3.1.3 Training loss

The WeblyNet system is trained using the following loss function.

$$\mathcal{L}(X_1, X_2, y) = \mathcal{L}(\mathcal{N}_1(X_1), y) + \mathcal{L}(\mathcal{N}_2(X_2), y) + \alpha \cdot D(\mathcal{N}_1(X_1), \mathcal{N}_2(X_2)) \quad (1)$$

The first two terms are losses calculated from outputs of two networks with the target classes of the dataset. Since one instance can belong to multiple classes, the binary cross-entropy loss with respect

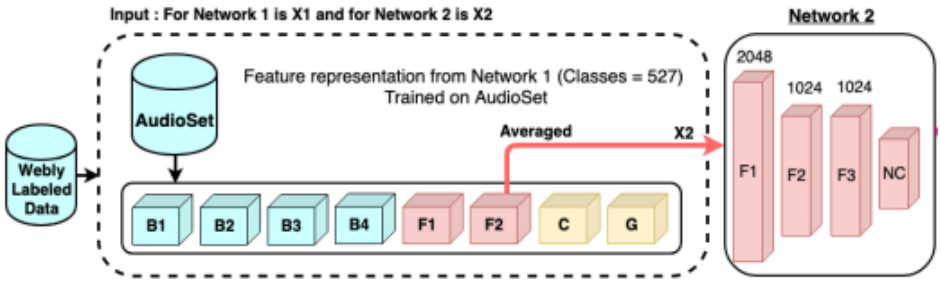


Figure 3: Network 2 Architecture Retrieved From Paper [11]

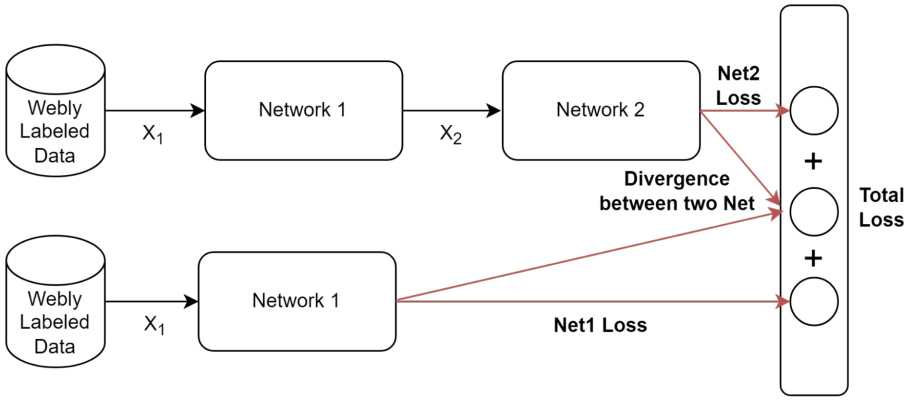


Figure 4: Network loss

to each class is applied. The overall loss of an instance is the mean of losses of all classes.

$$\mathcal{L}(\mathcal{N}(X), y) = \frac{1}{C}\sum_{c=1}^{C} l(y_c, p_c) \tag{2}$$

where $l(y_c, p_c) = -y_c * log(p_c) - (1 - y_c) * log(1 - p_c)$ and $p_c = \mathcal{N}(X)$

The final term measures divergence between outputs from 2 networks. They use combined generalized KL-divergences with respect to both outputs.

$$D(\mathcal{N}_1(X_1), \mathcal{N}_2(X_2)) = \sum_{i=1}^{C} (\mathbf{o}_i^{\mathcal{N}_1} - \mathbf{o}_i^{\mathcal{N}_2}) log \frac{\mathbf{o}_i^{\mathcal{N}_1}}{\mathbf{o}_i^{\mathcal{N}_2}} \tag{3}$$

In addition to the term, hyperparameter term α is applied to this term to control weight it contributes to the total loss.

3.2 WeblyNet System with Similarity Loss Function

In the prior work, they implicitly introduce independence to two networks by using different views of data so that these networks will not end up learning the same thing. This approach might include some degrees of dependence among networks.

We add an additional term to the total loss to force networks to learn different things. This term represents the similarity between pairs of different networks. We minimize this new term along with other losses during the training to force them to not learn same things and generate the same outputs. We also control the weight given to this similarity term in the total loss with an additional hyperparameter β. The new total loss we use in our proposed method is shown as follow:

$$\mathcal{L}(X_1, X_2, y) = \mathcal{L}(\mathcal{N}_1(X_1), y) + \mathcal{L}(\mathcal{N}_2(X_2), y) + \alpha \cdot D(\mathcal{N}_1(X_1), \mathcal{N}_2(X_2)) + \beta(Similarity) \quad (4)$$

In the study, we use multiple networks with the same architecture for simplicity. Then we use the proposed new term to force each models to learn different parameters and hidden representations of data for every layer.

We experiment this new total loss with candidates to be used for similarity term(s).

3.2.1 Cosine Similarity between models' parameters

In this approach, we will measure cosine similarity between parameters of two different models at different layers. The overall similarity is the mean of similarities over all layers. Suppose W_l^1 and W_l^2 are parameters at layer l from $\mathcal{N}_1$ and $\mathcal{N}_2$ respectively, the cosine similarity is $S_{cos}(W_l^1, W_l^2) = \frac{W_l^1 \cdot W_l^2}{||W_l^1|| ||W_l^2||}$. Then, the overall cosine similarity is

$$S_{cos}(\mathcal{N}_1, \mathcal{N}_2) = \frac{1}{L} \sum_{l=1}^{L} \frac{W_l^1 \cdot W_l^2}{||W_l^1||\ ||W_l^2||}$$

This overall cosine similarity is used as the similarity term in the total loss of the WeblyNet system.

3.2.2 Centered kernel alignment similarity between learned representations from layers of different models

Centered kernel alignment (CKA) is a tool proposed by Kornblith, et al.[8] for quickly comparing two representations in a way that is invariant to invertible linear transformation, orthogonal transformation, and isotropic scaling. Kornblith[8] introduced CKA to overcome the problem of previous similarity measurements such as CCA and its variants (Raghu et al., 2017[14]; Morcos et al., 2018[13]) that cannot compute similarity in the case which number of dimensions are higher than data points.

We will adapt this concept and calculate CKA similarity between two representations from layers of two different models. The overall similarity is the mean of similarities over all layers. If Z_l^1 and Z_l^2 are hidden representations at layer l from $\mathcal{N}_1$ and $\mathcal{N}_2$ respectively, then the CKA similarity is

$$CKA(Z_l^1, Z_l^2) = \frac{||(Z_l^1)^T (Z_l^2)||_F^2}{||(Z_l^1)^T (Z_l^1)||_F\ ||(Z_l^2)^T (Z_l^2)||_F}$$

Thus, the overall CKA similarity is $\sum_{l=1}^{L} CKA(Z_l^1, Z_l^2)$. This overall CKA similarity is use as the similarity term in the total loss of the WeblyNet system. In this study, we implement CKA using source code from `https://github.com/moskomule/anatome+`[6] which implements CKA based on the work of Kornblith, et al.[8].

3.3 WeblyNet system with modified networks divergence loss

As mentioned in the baseline model section, they measure pairwise divergence between outputs from two networks with symmetrical KL divergence for all coordinates of the output vectors. Although this can enable models to co-teach each other, it can also lead to different networks learning the same thing and generating same outputs. As a result, we will lose the ensemble prediction power of the whole network since we want them to learn different aspects of the data.

To eliminate this effect, we want the networks to co-teach each other only at coordinates of the positive target variables to make outputs from different networks at these positive positions close to each other. While for negative labels, we want the outputs from different networks to be as dissimilar as possible. Therefore, we modify the divergence term so that it gives different weights to positive and negative labels.

Let X_1 and X_2 are input for networks $\mathcal{N}_1$ and $\mathcal{N}_2$ respectively, and $\mathbf{o}_i^{\mathcal{N}_1}$, $\mathbf{o}_i^{\mathcal{N}_2}$ are outputs

from networks, and C_p and C_n are labels belong to positive classes and negative classes. The modified equation of divergence term is:

$$D(\mathcal{N}_1(X_1), \mathcal{N}_2(X_2)) = \sum_{i=1, i \in C_p}^{C} (\mathbf{o}_i^{\mathcal{N}_1} - \mathbf{o}_i^{\mathcal{N}_2}) log \frac{\mathbf{o}_i^{\mathcal{N}_1}}{\mathbf{o}_i^{\mathcal{N}_2}} + \gamma \sum_{i=1, i \in C_n}^{C} (\mathbf{o}_i^{\mathcal{N}_1} - \mathbf{o}_i^{\mathcal{N}_2}) log \frac{\mathbf{o}_i^{\mathcal{N}_1}}{\mathbf{o}_i^{\mathcal{N}_2}} \quad (5)$$

The first term is a divergence of outputs corresponding to positive labels, the second term is the divergence of outputs corresponding to negative labels with new factor γ, where $\gamma \in$ R. To enforce the outputs corresponding to negative labels to be dissimilar, we set γ to be a negative value, and we can config this factor as a hyperparameter of the system. With γ of 1, the divergence term is the same as we try to experiment with this new hyperparameter to see the effect of modified divergence term on the overall model performance.

To extend the use of this divergence term on more than two networks is simply perform the divergence for each and every pair of networks possible in the system. Given K networks in the system such that $K > 2$, we can measure $\frac{K(K-1)}{2}$ pairs of similarity.

4 Dataset description

The dataset AudioSet[4] and Webly-2k[7] were obtained from actual data used in previous work by Kumar, et al[1].

4.1 AudioSet

Audio Set, a large scale, 2.1M 10-second ontology of manually-annotated audio events and 527 classes is use in the experiments as input to train Network 1 to get the 1,024-dimension representation of the data.

4.2 Webly audio dataset

In previous work of Kumar el al.[11], they selected 40 vocabs of sounds from AudioSet as text search queries. The audio dataset was retrieved from YouTube videos, by querying the name of the sound, combined with the phrase "sound of". The Webly-2k audio dataset only use top 50 retrieved, around 1,900 audio recordings, from the above-mentioned text search query for each class. Only recordings under 4 minutes are considered. The data are multi-label since one video can be a result for multiple queries. The average length of Webly-2k data is at 111 seconds, a total of 60 hours. The Webly-2k audio is used to generate two views of data to feed into different networks.

Primary representation $\mathbf{X}_1$ The primary representation of webly data is the input for Network 1. This data consists of 128 embedding features of the audio recording provided by Google[7]. Those features represent quantized vectors for each 1 second of audio. Each file is of size N $\times$ 128, where N is the length of audio in seconds. The time-sequence of the audio is maintained by stacking the vector of 128 embedding features following the previous second.

Transformed representation $\mathbf{X}_2$ This representation is created by feeding primary representation X_1 to the pretrained Network 1 with another large-scale sound events dataset such as the AudioSet dataset, then extracts outputs from the F2 layer, and average the outputs over 1 second. This process provides 1,024 embedding features which will be used to train the Network 2.

5 Baseline and evaluation metrics

5.1 Reproduction of the baseline

We have reproduced the code, data and model implemented by Kumar, et al.[11] as a baseline with Mean Average Precision (MAP) of all classes as a metric. The training and validation dataset we used was the pre-processed data generated as the webly dataset (Webly-2k) in the text file format. They were splitted using the same criteria provided as a list. The testing dataset comes from AudioSet

Table 1: MAP comparison on different models

Method	MAP
$N_1 - Self(Baseline)(2k)$	38.0
$N_2 - Self(2k)$	41.2
WeblyNet (2k)	**44.0**

Table 2: Reproductions of baseline model experiments

Experiment detail	MAP
Test run (5 epochs)	4.9
Default setup (250 epochs)	42.5
Default setup (250 epochs); adjust learning rate to 0.001	41.8
Default setup (250 epochs); set numpy seed as 2019	42.9
Default setup (250 epochs); set torch, numpy, random seed as 2019	42.4
Default setup (250 epochs); set torch, numpy, random seed as 2019; set scheduler = 'Y'	42.5

which was also pre-processed as multiple text files. The pre-trained model was provided as the pickle file to perform feature embedding and feed to Network 2. We implemented the same model structure for Network 1 and Network 2, the same loss function and other arguments from the default setup. The original work MAP is shown in Table 1. The reproduction was done with several approaches shown in Table 2. The best MAP we reached is at 42.5, therefore we will use this MAP as the baseline.

5.2 Evaluation metric

During the inference, they use the average of outputs from Network 1 and Network 2 as the prediction of the whole system. The model's performance is assessed on the eval set of AudioSet, which contains around 4,500 recordings corresponding to the same 40 sound events in the Webly dataset. They use average precision (AP) to measure the performance of each class. For average precision (AP), precision and recall are calculated at different thresholds of the final output of each class.

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

The area under the curve between precision and recall is measured as average precision. The overall metric of the system is measured by mean average precision (MAP).

$$MAP = \sum_{i=1}^{C} AP_i$$

6 Experiments, results and discussion

6.1 Experiment results

To test the experiments, we setup the same environment to run the models as below

Dataset We use the same training, validation and test dataset which is already preprocessed and splitted in the text files format.

Hyperparameters We keep all hyperparameters in the model to be the same during the experiments, including: optimizer, scheduler, divergence factor, learning rate, number of epochs.

Table 3: Experiments on two different network settings and variants in hyperparameters

Experiment	Divergence	Similarity	Networks	β	MAP	MAUC
WeblyNet[1]	KL	-	N1 + N2	-	44.0	
Baseline implementation	KL	-	N1 + N2	-	42.6	
Baseline Network architecture 1.0	KL	-	N1 + N1	0	37.0	87.5
Variant 1.1	KL	Cosine	N1 + N1	0.005	37.1	87.4
Variant 1.2	KL	Cosine	N1 + N1	0.01	37.7	87.0
Variant 1.3	KL	Cosine	N1 + N1	0.05	37.5	87.4
Variant 1.4	KL	Cosine	N1 + N1	0.1	36.8	86.8
Baseline Network architecture 2.0	KL	-	N2 + N2	0	41.5	91.3
Variant 2.1	KL	Cosine	N2 + N2	0.005	41.4	91.2
Variant 2.2	KL	Cosine	N2 + N2	0.01	40.9	91.0
Variant 2.3	KL	Cosine	N2 + N2	0.03	41.3	91.0
Variant 2.4	KL	Cosine	N2 + N2	1	41.4	90.8
Variant 2.5	KL	Cosine	N2 + N2	2	41.2	91.0
Variant 2.6	KL	Cosine	N2 + N2	5	40.8	90.8
Variant 2.7	KL	CKA	N2 + N2	0.0001	41.3	91.1
Variant 2.8	KL	CKA	N2 + N2	0.001	41.3	91.3
Variant 2.9	KL	CKA	N2 + N2	0.01	41.5	91.0
Variant 2.10	KL	CKA	N2 + N2	0.1	41.5	91.0
Variant 2.11	KL	CKA	N2 + N2	1	41.2	91.1
Variant 2.12	KL	CKA	N2 + N2	10	41.1	91.0
Variant 2.12	KL	CKA	N2 + N2	100	41.1	91.1

Validation Process We run the models with the same number of epochs and choose the best model which has the highest validation MAP predicted on the test dataset.

We run two different approaches to test how they impact the baseline model and report the performance by MAP and MAUC on the test dataset.

Similarity approach We run the experiments by adding the similarity measurement as an additional term in the loss function to avoid two models to learn the same thing. The two similarity measurements we test are Cosine Similarity and Centered Kernel Alignment (CKA) in which we measure the distance between weights of the two similar network architectures. This setup is different from the model implemented in paper [11] which uses two networks with different architectures. Therefore, we first run two models with the same networks wish to test as a baseline model for this approach, and then test the impact of similarity by varying the similarity factor.

We experiments on 2 settings as follow:

1. Co-train Network 1 with Network 1 in which we measure the similarity distance by using only Cosine Similarity.
2. Co-train Network 2 with Network 2 in which we measure the similarity distance by using Cosine Similarity and Linear CKA[8].

Since the Linear CKA measurement supports only the Linear model, we test the CKA model only on Network 2 which consists only of layers of Linear.

From Table 3, we can see that in case of two CNN networks (N1+N1) with cosine similarity as similarity term, MAP slightly increases from the baseline as we increase β factor. But, after some point MAP starts to drop. It is possible that when we enforce exceeding dissimilarity between networks, the performance of some models might drop so that overall performance drops as well. However, the improvement in this setting is not large enough to conclude that cosine similarity terms can rise MAP. In the case of two FC networks (N2 + N2) with cosine similarity, the performance seems to get worse or not different than the baseline.

Table 4: Experiments on different network settings, masking, and variants in hyperparameters

Experiment	Divergence	Networks	γ	Best epoch	MAP	MAUC
WeblyNet[1]	KL	N1 + N2	-	-	44.0	
Baseline implementation	KL	N1 + N2	-	-	42.6	
Baseline Network architecture 2.0	KL	N1 + N2	1	95	42.6	90.9
Variant 2.1	Masked KL	N1 + N2	0	64	42.1	90.6
Variant 2.2	Masked KL	N1 + N2	-0.001	48	42.0	90.7
Variant 2.3	Masked KL	N1 + N2	-0.0001	101	41.9	90.8
Variant 2.4	Masked KL	N1 + N2	-0.00001	64	42.4	91.0
Variant 2.5	Masked KL	N1 + N2	-0.000001	92	41.9	90.8
Variant 2.6	Masked KL	N1 + N2	0.001	64	41.9	90.8
Baseline Network architecture 3.0	KL	N1 + N1 + N2	1	128	42.6	90.9
Variant 3.1	Masked KL	N1 + N1 + N2	-0.001	105	42.9	91.4
Variant 3.2	Masked KL	N1 + N1 + N2	-0.0001	93	42.9	91.2
Variant 3.3	Masked KL	N1 + N1 + N2	-0.00001	184	43.1	91.4
Variant 3.4	Masked KL	N1 + N1 + N2	-0.000001	103	43.6	91.3
Baseline Network architecture 4.0	KL	N1 + N1 + N1 + N2	1	192	42.0	90.5
Variant 4.1	Masked KL	N1 + N1 + N1 + N2	-0.001	112	42.2	90.9
Variant 4.2	Masked KL	N1 + N1 + N1 + N2	-0.0001	159	42.5	91.3
Variant 4.3	Masked KL	N1 + N1 + N1 + N2	-0.00001	102	43.1	91.3
Variant 4.4	Masked KL	N1 + N1 + N1 + N2	-0.000001	109	42.4	91.1

From these two cases, we cannot conclude that cosine similarity can boost the performance of the whole network. One possible reason is that cosine similarity cannot measure similarity between parameters of the networks that are invariant to linear transformation or neuron relocation since there is a chance that two networks are the same with neurons located in different coordinates of the parameters matrices.

In the case of CKA as similarity measure, we can also see from the table that the performance seems to not different than the baseline, and starts to get worse when we keep increasing factor. This might be because the CKA term contradicts the existing network divergence term. When we increase the effect of the CKA term with β, the whole system loses the effect of co-teaching between networks.

Divergence modification approach We run the experiments by modifying the divergence term in the existing loss function to multiply it by the masked value which we calculated from the negative label derived from Equation (1). We first run the baseline model which has the same architecture as the paper [11] and vary the masked value to test the impact on the model performance. Then we introduce more networks to the model, find the baseline without modifying the existing KL divergence term, then vary the masked value to test the impact to the model. We test with 2, 3, and 4 networks and summarize the MAP and MAUC on the test dataset.

Table 4 shows that final performances of two networks system with modified divergence term are worse than the baseline model. But when we investigate more on plots between validation MAP and training epoch in Figure 5a, we can see that models with modified divergence terms converge much faster than the original divergence term. However, the modified term makes models become overfitting much faster as well. As a result, the final performance that the baseline model achieve is higher.

For the system with 3 networks (2 CNN networks and 1 FC network), unlike the two networks system, the final performance that models with the modified divergence term tends to be higher than the baseline network. According to Figure 5b, similarly to the two networks system, we can see that models with modified divergence term converge much faster, but now as we add more network to the system, the variance of the model decrease. As a result, the final models become less overfit and final performances are higher.

Table 5: Performance comparison of each experiments to the baseline, no modifications model

Experiment	Networks	MAP	Δ (baseline)	Δ (WeblyNet[1])
Cosine similarity (Network 1)	N1 + N1	37.5	+0.5	-5.2
Cosine similarity (Network 2)	N2 + N2	41.4	-0.1	-1.2
CKA (Network 2)	N2 + N2	41.5	+0.0	-1.1
Masked KL divergence (2 networks)	N1 + N2	42.4	-0.3	-0.3
Masked KL divergence (3 networks)	N1 + N1 + N2	43.6	+1.0	+1.0
Masked KL divergence (4 networks)	N1 + N1 + N1 + N2	43.1	+1.1	+0.4

For the system with 4 networks (3 CNN networks and 1 FC network), similar to 3 networks system, with modified divergence term, the model converges faster and final performance is higher than the baseline model.

6.2 Components contribution

We summarize the best performance for each experiment and compare it with the baseline which has the same network architecture and baseline from WeblyNet[11].

As a result, KL divergence modification with masked value on the negative label on 3 networks has the highest MAP on the test dataset. It increases the MAP from the baseline with the same network by +1.0 and also increases from WeblyNet [11] by +1.0.

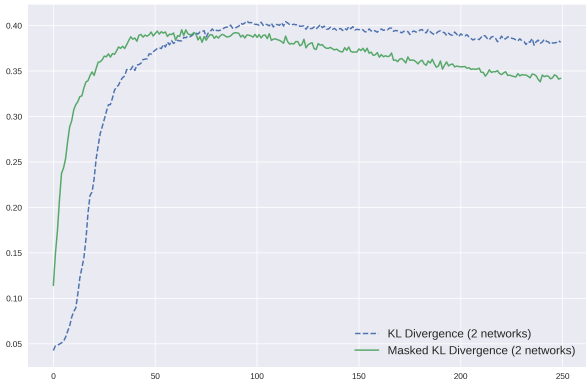
According to all experiments, we can conclude that additional similarity terms we implemented do not contribute much to the improvement of final performance due to their bad compatibility with existing terms. For modified KL divergence, it helps improve the convergence of the model, but it makes the system more overfitted. However, we can overcome this disadvantage by integrating more networks into the system. As a result, the modified term works better with more networks in the system, inspected in Figure 5.

7 Conclusion

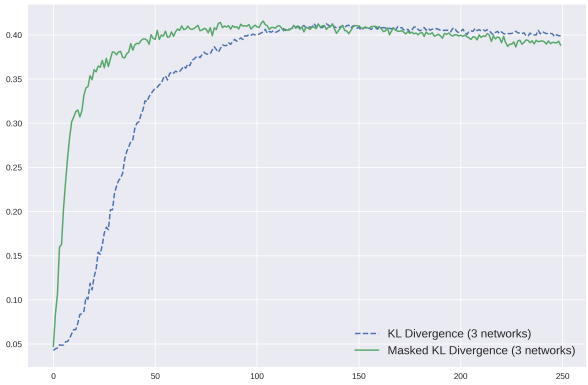
In this project, we experiment with the methods to improve the performance of the WeblyNet system on the Webly data by trying to introduce more independence between sub-networks in the system. We incorporate an additional similarity term to overall training loss to force the model to learn different aspects of the data. We explore two candidates of this similarity term which are cosine similarity between parameters of a pair of sub-networks and centered kernel alignment (CKA). Furthermore, we experiment with another approach by modifying the network outputs divergence term used in the previous work [11] to have different weights given to outputs corresponding to positive labels and negative labels.

Initially, we hypothesized that minimizing the similarity term that we add to the overall train loss could help increase the degree of independence among different sub-networks so the ensemble system could overcome the noisiness of Webly data and improve its performance. However, results show that both candidates for the additional term do not contribute to the improvement of the whole system. This might be because of the fact that the similarity term between networks' parameters contradicts the existing divergence loss term. Therefore, the overall performance does not change much and gets worse given the large similarity factor. We conclude that the additional similarity loss term does not work well with the existing setting of training loss. Hence, further experiments with different combinations of different divergence terms and different similarity terms need to be explored.

On the other hand, we hypothesized that the existing network outputs divergence loss term can lead to different networks learning the same thing and then generating the same outputs. After conducting the experiment, results show that convergence of the whole network is faster with modified network divergence terms with different weights given to different outputs corresponding to positive and negative labels. Although this can also lead to overfitting, we can overcome this by adding more networks to the model. And the overall performance of the whole network improves with the modified network divergence terms. However, the masking factor in the modified term needs to be tuned as a



(a) Trained with 2 networks



(b) Trained with 3 networks

Figure 5: MAP of KL divergence and masked KL divergence

hyperparameter to achieve the highest benefit. Therefore, we conclude that the modified network output divergence term can be a way to improve the overall performance of the WeblyNet system.

8 Proposed extensions

Our future works include the following:

1. Explore different combinations of different network output divergence loss terms and similarity terms. As we conclude that the similarity term contradicts with existing divergence term in the training loss, we want to explore more other candidates for output divergence term that are more compatible with the similarity term such as CKA. The first option that we can experiment with is the modified output divergence term that we use in this study.
2. Experiment on similarity terms between different network architectures. In this study, we implement similarity loss term in training loss of webly system that consists of multiple networks with the same architecture due to limitation of cosine similarity and CKA. We want to explore other methods that can measure learning similarity between different architectures. We can design architectures to have an identical layer such as the layer before the output layer, we can measure the similarity between the hidden representations of data generated from this specific layer of different networks.

3. Explore the effect of modified output divergence terms on the different data types. In this study, we use sound data and webly labeled data obtained from the internet to train webly system with modified output divergence terms. We want to see the effect of the term on different data types such as image, video, or text data.

References

[1] Xinlei Chen and Abhinav Gupta. Webly supervised learning of convolutional networks. *CoRR*, abs/1505.01554, 2015.

[2] Haytham M. Fayek and Anurag Kumar. Large scale audiovisual learning of sounds with weakly labeled data, 2020.

[3] Rob Fergus, Li Fei-Fei, Pietro Perona, and Andrew Zisserman. Learning object categories from internet image searches. *Proceedings of the IEEE*, 98(8):1453–1466, 2010.

[4] Jort F. Gemmeke, Daniel P. W. Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R. Channing Moore, Manoj Plakal, and Marvin Ritter. Audio set: An ontology and human-labeled dataset for audio events. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 776–780, 2017.

[5] Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor W. Tsang, and Masashi Sugiyama. Co-sampling: Training robust networks for extremely noisy supervision. *CoRR*, abs/1804.06872, 2018.

[6] Ryuichiro Hataya. anatome, a pytorch library to analyze internal representation of neural networks, 2020.

[7] Shawn Hershey, Sourish Chaudhuri, Daniel P. W. Ellis, Jort F. Gemmeke, Aren Jansen, R. Channing Moore, Manoj Plakal, Devin Platt, Rif A. Saurous, Bryan Seybold, Malcolm Slaney, Ron J. Weiss, and Kevin Wilson. Cnn architectures for large-scale audio classification, 2016.

[8] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited, 2019.

[9] Anurag Kumar. *Acoustic Intelligence In Machines*. PhD thesis, 09 2018.

[10] Anurag Kumar, Maksim Khadkevich, and Christian Fügen. Knowledge transfer from weakly labeled audio using convolutional neural network for sound events and scenes. *CoRR*, abs/1711.01369, 2017.

[11] Anurag Kumar, Ankit Shah, Alexander G. Hauptmann, and Bhiksha Raj. Learning sound events from webly labeled data. *CoRR*, abs/1811.09967, 2018.

[12] Yingming Li, Ming Yang, and Zhongfei Zhang. Multi-view representation learning: A survey from shallow methods to deep methods. *CoRR*, abs/1610.01206, 2016.

[13] Ari S. Morcos, Maithra Raghu, and Samy Bengio. Insights on representational similarity in neural networks with canonical correlation, 2018.

[14] Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability, 2017.