MLP Coursework 3: Self-paced Learning

s1646851,s1702319 ,s1752655

Abstract

Self-paced learning (SPL) is a recently proposed learning strategy which can effectively avoid bad local optimum in neural networks. Inspired by the learning process of humans and animals, the technique literately incorporates easy to more complex samples into training. One of the advantages of SPL is that it can build different self-paced regularization term for diverse datasets, which are used to define easy examples. For instance, traditional SPL regards easy examples as those which have smaller training loss or higher likelihood. In this project, we aim at exploring self-paced learning technique on the CIFAR-10 dataset and find out which self-paced regularization terms outperform others for this particular dataset. In this report, we list some related works and build a 2-layer convolutional neural network as baseline.

1. Motivation

Humans and animals start learning from simple and basic concepts and then gradually learn complex ones in an organized way. This idea was called *shaping*, which can be widely used in training animals. The term was coined in Skinner (1958), which was inspired from animals' innate repertoires of responses that to some extent, the aim of learning easy and simple tasks is to provide help to complex and difficult tasks (Breland & Breland, 1961).

Now shaping has been widely used, not only in animal training. Many researchers focus on developing machine learning strategies according to human study models and intend to figure out whether this kind of strategy is able to perform better. The idea that training a learning machine using similar strategy could go back to Elman (1993), which emphasized the importance of starting small. He states that training should start with an easier task, then gradually improve the difficulty and complexity, just similar to human trained with a curriculum. According to the results of the simulations with artificial neural network conducted by Elman (1993), a network appears to do better at learning the grammatical structure when it begins with a set of restricted data and then has incremental input and incremental memory. However, Rohde & Plaut (1999) put out different ideas from Elman. They found that the exposure to the full complexity of the language makes the network learn the language most effectively in training. The layer structure of the deep neural network is a little bit analogical to the human learning process. For example, the first layer deals with simple abstractions, but along with the gradual increase of layers, the deepness of abstraction also increases which means that learning grows hard. In 2009, Krueger & Dayan (2009) found that shaping as a method of training offers significant benefits for learning from the cognitive perspective. Based on these previous research, it is necessary to study further on starting small strategy, which is also called *curriculum strategy* to find out how it benefits machine learning problems. Bengio et al. (2009) published *curriculum learning* in 2009, in which they introduced this strategy in detail.

Curriculum learning strategy is committed to breaking through the limitations of deep neural networks: non-convex optimization problems (help to find a better local minimum and speed up the convergence of training towards the global minimum). Solving a non-convex optimization task is an essential task in the machine learning literature. The central hypothesis they raised in the paper was that 'a well-chosen curriculum strategy could act as a continuation method.' According to Allgower & Georg (2012), continuation methods are optimization strategies to finds best local minima of non-convex criteria. 'Applying a continuation method to the problem of minimizing a training criterion involves a sequence of training criteria, starting from one that is easier to optimize, and ending with the training criterion of interest.' Furthermore, they also stated that 'a curriculum strategy can speed up the convergence of training towards the global minimum.' In all, curriculum learning has positive influences on boosting algorithms. Curriculum learning strategies are also considered as a particular form of transfer learning. The initial tasks are used to guide the learners to have better performance on finishing the final tasks.

One of the key questions in this strategy is how to judge what is easy. Bengio et al. (2009) state that it depends, and then did three experiments with a convex criterion and experiments on shape recognition, as well as on language modeling conducted to verify this idea.

Through several experiments in different settings, they gave a positive answer to the question left by previous cognitive science research, which is whether a curriculum strategy can have positive effects on machine learning algorithms. After answering this question, they continued to propose several hypotheses to explain the potential advantages of this strategy, including "faster training in the online setting and guiding training towards better regions in parameter space".

Although curriculum learning could find better local min-

ima for non-convex criterion and improve the speed of training, its drawback is obvious. A main challenge for curriculum learning is that it is difficult to identify easy tasks and complex tasks to the algorithm automatically in many real-world applications (Kumar et al., 2010) and lack of generalization (Jiang et al., 2014). A curriculum for a model is set before training and keeps fixed over the whole process of training and determined by common sense of the participants. Therefore, the setting of a curriculum is independent with the subsequent learning, which could cause that the fixed curriculum is inconsistent with dynamically learned models (Jiang et al., 2015).

Spitkovsky et al. (2009) approached grammar induction, where the ranking function is derived in terms of the length of a sentence. The heuristic is beneficial in solving these kinds of problems since short sentences are easier to learn and should be learned earlier and that along with the increase of the length of sentences, the possible solutions also increase. The heuristical curriculum, however, has a main disadvantage. Since the curriculum has been decided and fixed in advance, it is difficult to change it according to the feedback and adjust it to learners dynamically.(Jiang et al., 2015) There is also another challenge for curriculum learning which has been mentioned before. Machines cannot judge what is sample and what is difficult by themselves. Moreover, the form of easy sample depends on different trainings.

Faced with these challenges, Kumar et al. (2010) presented an idea called *self-paced learning*(SPL). In general, what SPL do is like let students judge what is easy versus complex by themselves rather than told by teachers (Supancic & Ramanan, 2013). In SPL, the introduction of the term regularization into the learning objective helps models learn both easy and complex samples together and the sequence of a curriculum is gradually determined by the model itself based on what it has already learned, rather than predefined heuristic criteria. SPL can be used for general realization of curriculum learning because it is not designed to solve problems in specific model objectives.

In the context of a latent variable model, for a given parameter $\mathbf{w}$, this easiness can be defined in two ways: (i) a sample is easy if we are confident about the value of a hidden variable; or (ii) a sample is easy if it is easy to predict its true output. The two definitions are somewhat related: if we are more certain about the hidden variable, we may be more certain about the prediction. They are different in that certainty does not imply correctness, and the hidden variables may not be directly relevant to what makes the output of a sample easy to predict. Kumar et al. (2010) therefore focus on the second definition: easy samples are ones whose correct output can be predicted easily (its likelihood is high, or it lies far from the margin).

To verify the usability of the SPL, Kumar et al. (2010) did experimentes which were aimed to solve a biconvex optimization problem. They used four standard datasets from disparate domains (natural language processing, computational biology and computer vision). Their method performed better than others in all these datasets.

In human education, self-paced learning means that curriculum design is dependent on the ability of learners rather than fixed in advance by the instructor(Jiang et al., 2015). Compared with curriculum learning, self-paced learning has two advantages. Firstly, it combines the learning objectives with the curriculum so that the curriculum and the learned model are united in the same optimization problem. Secondly, it separates the regularization term from the loss of functions of specific problems.

This strategy has been successfully used in various applications, such as action/event detection(Jiang et al., 2012), domain adaption(Tang et al., 2012a), dictionary learning (Tang et al., 2012b), tracking(Supancic & Ramanan, 2013), segmentation(Kumar et al., 2011) and face identification(Lin et al., 2018).

2. Research questions

In this project, we aim at exploring self-paced learning technique on the CIFAR-10 dataset and find out which self-paced regularization terms outperform others for this particular dataset. It has been demonstrated that the self-paced learning technique outperforms the state of the art methods in object localization, noun phrase coreference, motif finding and handwritten digit recognition (Kumar et al., 2010). Therefore, we'd like to run experiments to find out whether self-paced learning algorithm will improve classification accuracy on the CIFAR-10 dataset. Image classification has become one of the most important research areas in the field of computer version in recent years. By self-paced learning, this report focuses on the process of learning from easy samples to complex ones, which can also solve the non-convex optimization problem. In self-paced learning, we tend to choose examples which have high likelihood or small training loss in each iteration.

3. Objectives

Our main objective is to improve classification accuracy on the CIFAR-10 dataset using self-paced learning technique. Moreover, if things go well we plan to explore different ways of defining easy and complex examples in our experiment.

In the further experiments, we will apply the self-paced learning to the baseline, and we expect this method could improve the performance of the model.

4. Data

The task of this project is to explore the classification of images using convolutional neural networks. We will use loss and accuracy rate to evaluate this task

In this project we will use the CIFAR-10 dataset, http://www.cs.toronto.edu/~kriz/cifar.html. The

CIFAR-10 is labelled subsets of the 80 million tiny images dataset. It was collected by Krizhevsky & Hinton (2010). The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes *airplane*, *automobile*, *bird*, *cat*, *deer*, *dog*, *frog*, *horse*, *ship and truck*, with 6000 images per class. There are 50000 training images and 10000 test images. The dataset is divided into five training batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly-selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class. Figure 1 shows a random sample of images from this dataset. Each image in the dataset has a very noisy label, which is the search keyword used to find it. The classes are designed to be completely mutually exclusive. For example, neither automobile nor truck contains images of pickup trucks.

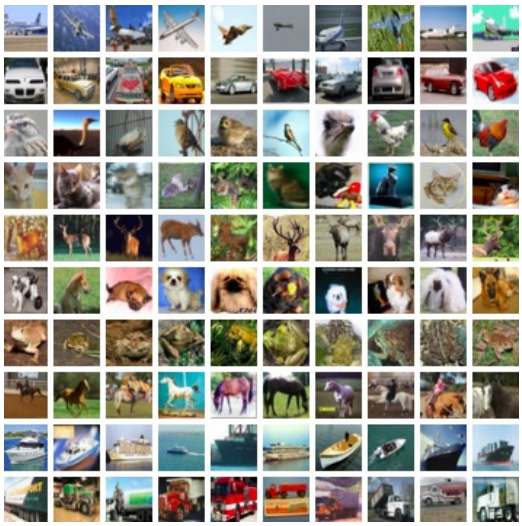

Figure 1. a random sample of images from CIFAR-10 dataset.

The images dataset is pre-processed by deleting the 1000 least-significant principal components. The pre-processed images look identical to the original images to the naked eye. Krizhevsky & Hinton (2010) also subtract the mean and set the average standard deviation over all dimensions to 1.

5. Methodology

In curriculum learning, a *curriculum* is a ranked list of training sets that is consistent with a sequence of the samples sorted in an ascending order of learning difficulty (Bengio et al., 2009). The basic idea of Curriculum Learning(CL) is to sort the subtasks by difficulty, begin with simple subtasks, and then enhance the difficulty level. The easy samples mean the samples are more noiseless; oppositely the complex samples are the samples are more noisy, in which case it's impossible to exactly predict the output from input samples.

The key of the curriculum is to create a ranking function that evaluates the difficulty level of each training sample and assigns training priorities. Bengio et al. (2009) shows a statistical definition of a curriculum that evaluates the difficulty degree from 'easy' to 'complex' based on a list of distributions and the entropies of those distributions should be an ascending order.

Inspired by Continuation methods (Allgower & Georg, 1980) that are optimization methods for solving the minimizing non-convex problem, Bengio et al. (2009) states that a curriculum also is an ordered list of training criterion. Each training criterion in this list is correlative with a different series of weights on the training samples, in other words, on a reweighting distribution of training data. At first, the weights prefer to 'simpler' training samples, or samples explaining the simpler concept, which can be learned easier. The following training criterion can change slightly the weights of features that increase the probability of more complex examples.

This idea also can be formalized. Assume z be a random variable which represent an example for the model (specially, an (x,y) pair for supervised learning). Assume $P(z)$ be the target training distribution that the model should eventually learn. Assume $0 \leq W_\lambda(z) \leq 1$ be the weights for example z at step λ in the curriculum list. The relevant training distribution is

$$Q_\lambda(z) \propto W_\lambda(z)P(z) \qquad \forall z \tag{1}$$

Definition 1 *Bengio et al. (2009) defines the relevant list of training distribution Q_λ is a* ***curriculum****.*

Regarding distribution, the sum of all samples z value is 1:

$$\int Q_\lambda(z)dz = 1 \tag{2}$$

Then we have:

$$Q_1(z) = P(z) \quad \forall z \tag{3}$$

At the same time, with the increase of step λ (the sample size also increases), this strategy needs to meet two requirements:

Increasing entropy to increase the diversity of training samples:

$$H(Q_\lambda) < H(Q_{\lambda+\epsilon}) \quad \forall \epsilon > 0 \tag{4}$$

Increasing the weight of the sample in the training set so as to increase the number of samples:

$$W_{\lambda+\epsilon}(z) \geq W_\lambda(z) \quad \forall z, \forall \epsilon > 0 \tag{5}$$

Now, we consider a simple training set where the number of examples of Q_λ is finite, and increasing λ represents adding additional samples to this set. Assume we have a training set $D = \{(x_i, y_i)\}_{i=1}^n$, where (x_i, y_i) represent the i^{th} training data pair. The data pairs are ranked by entropy. A pair with a higher rank should be learned earlier.

5.1. Three curriculum learning strategies

How to derive the curriculum is the key for training a model with curriculum learning. Shi et al. (2015) introduced three curriculum learning methods, which all present the RNNLMs from general data to specific data. These CL methods gradually provide more effect for training data at the process of training models.

1. **Start-from-Vocabulary (Start-Vocab)** : This kind of curriculum learning makes use of the whole training sets to build the vocabulary space, while only uses the part of the training data from the relevant sub-domain for training.

2. **Data Sorting (Data-Sort)**: Unlike Start-Vocab, the whole training data, in this curriculum learning, is used to train the sub-domain-adapted RNNLMs. Before training, the training data is ranked. The sorted sequence of data is from the target sub-domain, from witch we select validation data. When the generalization of the model cannot be improved on the selected validation data, the learning rate of this model will reduce by half. Until the reduction of learning rate no more help optimize the model on the validation, we will stop the model training.

3. **All-then-Specific (All-Specific)**: In the all-specific curriculum learning, sub-domain-adapted models move from learning the entire training data at the majority of epochs (called general training period) to training on its specific sub-domain data (called specific training period). The validation data in the general training period is named the general validation data which must cover all the sub-domain; the validation data in the specific training period is named the specific validation data that is chosen from corresponding sub-domain of the model. The model training starts from the general training until halving the learning rate no longer help improve the performance of the model on the general validation data; then, the model training enters the specific training period and training in this period will be stopped when the reduction of learning rate cannot improve the generalization on the specific validation data.

Comparing three curriculum learning strategies, the Start-Vocab strategy only uses the data from the corresponding sub-domain. For Start-Vocab strategy, using the least general information results in small probabilities for the words in the sub-domain. However, both the methods of the Data-Sort and the All-Specific use the entire information of the training sets. The main difference of the two methods is the time when they move from the general training period to the specific training period. In the Data-Sort strategy, the model is trained to achieve a domain-specific minima of loss in each epoch. However, in the All-Specific strategy, the optimization of the model moves from the general training period to the specific training period.

5.2. Self-Paced Learning

The basic idea of the SPL is to design curriculum as a regularization term and add it to loss function. Therefore, the method of SPL has two main benefits: first, this method both optimizes the model and the curriculum at the same time; second, the regularization term cannot affect the optimization of weights in this model, because the self-paced regularization term is independent with the loss function.

The authors claim that given training set $\mathcal{D} = (\mathbf{x}_i, y_i), ..., (\mathbf{x}_n, y_n)$, the number of samples is n. Each $\mathbf{x}_i \in R^m$ represents a sample and y_i represents the classification label for the corresponding sample. The parameters $\mathbf{w}$ represent the optimization parameters of the corresponding model and its regularization item is $r(\mathbf{w})$. Then, They denote loss function as $f(x_i, y_i, w)$. So, an objective function of traditional machine learning algorithm can be denoted as:

$$\mathbf{w}_{t+1} = argmin(r(\mathbf{w}) + \sum_{i=1}^{n} f(x_i, y_i, \mathbf{w})) \tag{6}$$

The core idea of self-paced learning is to select samples with small training errors and high likelihood values from all samples in each iteration, and then, update the model parameters(Kumar et al., 2010). In each iteration, choosing the number of samples is determined by weighting parameters which can import more samples by decreasing itself. At last, it stops the iteration when all samples have been selected, or when the loss function is unable to decrease. Therefore, self-paced learning imports binary variable $\mathbf{v} = [v_1, ..., v_n]^T$ into traditional machine learning objective functions to represent whether each sample is selected, or whether it is a simple sample. For example, a simple sample is a sample that is high in likelihood or far away from the classification boundary. So, only the samples with $v_i = 1$ are included in the objective function calculation, and the objective function can be changed to:

$$(\mathbf{w}_{t+1}, \mathbf{v}_{t+1}) = argmin(r(\mathbf{w}) + \sum_{i=1}^{n} v_i f(x_i, y_i, \mathbf{w}) - \frac{1}{K} \sum_{i=1}^{n} v_i) \tag{7}$$

In this equation, K represents the parameter of self-paced learning which is used to decide which samples can be selected for training. If the K value is large, the objective function optimization process tends to choose a sample with a small loss value. As the number of iterations increases, they gradually reduce the value of K. As K values approaches to 0, more samples are selected. At this point, the self-paced learning training process is degenerated into the traditional machine learning training process. Therefore, the process of self-paced learning can be understood as starting with a small number of "simple" samples and gradually introducing more samples until all samples are selected into the model.

The algorithm 1 introduces pseudocode of self-spaced learning. In this algorithm, we need to initialize model parameters $\mathbf{w}$, and then update latent weights variable $\mathbf{v}$ as well as model parameters $\mathbf{w}$. In this process of update, we need to decrease K to add more training samples for training.

Algorithm 1 Self-spaced learning

Require: training sets $\mathcal{D}$
Ensure: model weights $\mathbf{w}$
initialize w^*
while not converged **do**
  while not converged **do**
    Update $\mathbf{v}^* = argmin_{\mathbf{v}} E(\mathbf{w}^*, \mathbf{v})$
    Update $\mathbf{w}^* = argmin_{\mathbf{w}} E(\mathbf{w}, \mathbf{v}^*)$
  end while
  decrease K
end while
return $w = w^*$

The basic self-paced learning optimization process is simple. $v_i = 1$ when $f(x_i, y_i, \mathbf{w}) \leq 1/K$. $v_i = 0$ when $f(x_i, y_i, \mathbf{w}) > 1/K$. In particular, the optimization problem of the above self-paced learning can be transformed into biconvex optimization problem when $f()$ and $r()$ are convex functions. The problem of biconvex optimization is that the parameter set z can be divided into two mutually exclusive sets z_1 and z_2. If any parameter set z_i fixed a set of parameter values, another set of parameter optimization problems can be seen as convex optimization problems. This kind of problem can be considered as a biconvex optimization problem. For example, in the above expressions, there are two sets of parameters $\mathbf{w}$ and $\mathbf{v}$. We can alternately fix $\mathbf{w}$ to obtain the optimal solution of $\mathbf{v}$, and then fix $\mathbf{v}$ to find the optimal solution of $\mathbf{w}$. This optimization method can be called alternative convex search (ACS) method, which can guarantee the local optimal solution of the function(Gorski et al., 2007).

6. Experiments

In this section, we will discuss baselines experiments using convolutional neural networks trained on CIFAR10. In our experiments, there is a total of 50000 images in both training sets, and 10000 test images. The hyper-parameters are the following: the depth of network, activation function, learning rate of adam optimizer, the probabilities of dropout, the use of batch normalization and the kinds of pooling layers. The selection of hyper-parameters is simplified using the following heuristic: all hyper-parameters were chosen so as to have the best baseline performance on validation set without self-space learning. These hyper-parameter values are then used for the self-space learning experiments. The details of those hyper-parameter values are shown in table1:

Depth	Activation Function	Optimizer	Learning Rate
2 layers	leaky_relu	adam	1e-3
Dropout	Batch Norm	Batch Size	Pool
50%	True	128	max pooling

Table 1. The hyper-parameter values of baseline

Most of the precise values of these of these hyperparameters are not terribly important, but we found these to work well on a validation set. The one important value is the learning rate. We found that setting this value too high caused the model to simply memorize the training data quickly, while too low learning rate would cost training time.

The Table 2 lists the performance of the baseline,

training loss	training accuracy	validation loss	validation accuracy
0.144	0.948	0.633	0.838

Table 2. The result of baseline

In the further experiments, we will add the technique of self-paced learning to the baseline, and we expect this method could improve the performance of the classifier.

7. Interim conclusions

So far, we have built 2-layer convolutional neural network as baseline which achieves 0.838 validation accuracy. After reading recent related works, we find that SPL technique does improve performance of neural networks in object localization and handwritten digit recognition. Moreover, it can enhance the robustness of neural networks by avoiding local optimum.

However, till now it is hard to say which self-paced regularization term in SPL is perfect for this particular CIFAR-10 dataset because of lack of further experiments.

8. Plan

Next, we will run experiments of different self-paced regularization term in SPL to find out the most suitable one for the CIFAR-10 dataset based on classification accuracy. Self-paced regularization term is always used to describe the easiness of examples in dataset. We plan to try several kind of regularization term represented in (Kumar et al., 2010; Jiang et al., 2014; Zhang et al., 2017) respectively. The original SPL regards easy examples as those which have smaller training loss or higher likelihood, while more recent techniques attach great importance to the diversity between different subsets and thus can avoid data sparsity.

References

Allgower, Eugene L and Georg, Kurt. *Numerical continuation methods: an introduction*, volume 13. Springer Science & Business Media, 1980.

Allgower, Eugene L and Georg, Kurt. *Numerical continuation methods: an introduction*, volume 13. Springer Science & Business Media, 2012.

Bengio, Yoshua, Louradour, Jérôme, Collobert, Ronan, and Weston, Jason. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pp. 41–48. ACM, 2009.

Breland, Keller and Breland, Marian. The misbehavior of organisms. *American psychologist*, 16(11):681, 1961.

Elman, Jeffrey L. Learning and development in neural networks: The importance of starting small. *Cognition*, 48(1):71–99, 1993.

Gorski, Jochen, Pfeuffer, Frank, and Klamroth, Kathrin. Biconvex sets and optimization with biconvex functions: a survey and extensions. *Mathematical Methods of Operations Research*, 66(3):373–407, 2007.

Jiang, Lu, Hauptmann, Alexander G, and Xiang, Guang. Leveraging high-level and low-level features for multimedia event detection. In *Proceedings of the 20th ACM international conference on Multimedia*, pp. 449–458. ACM, 2012.

Jiang, Lu, Meng, Deyu, Yu, Shoou-I, Lan, Zhenzhong, Shan, Shiguang, and Hauptmann, Alexander. Self-paced learning with diversity. In *Advances in Neural Information Processing Systems*, pp. 2078–2086, 2014.

Jiang, Lu, Meng, Deyu, Zhao, Qian, Shan, Shiguang, and Hauptmann, Alexander G. Self-paced curriculum learning. In *AAAI*, volume 2, pp. 6, 2015.

Krizhevsky, Alex and Hinton, G. Convolutional deep belief networks on cifar-10. *Unpublished manuscript*, 40:7, 2010.

Krueger, Kai A and Dayan, Peter. Flexible shaping: How learning in small steps helps. *Cognition*, 110(3):380–394, 2009.

Kumar, M Pawan, Packer, Benjamin, and Koller, Daphne. Self-paced learning for latent variable models. In *Advances in Neural Information Processing Systems*, pp. 1189–1197, 2010.

Kumar, M Pawan, Turki, Haithem, Preston, Dan, and Koller, Daphne. Learning specific-class segmentation from diverse data. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pp. 1800–1807. IEEE, 2011.

Lin, Liang, Wang, Keze, Meng, Deyu, Zuo, Wangmeng, and Zhang, Lei. Active self-paced learning for cost-effective and progressive face identification. *IEEE transactions on pattern analysis and machine intelligence*, 40 (1):7–19, 2018.

Rohde, Douglas LT and Plaut, David C. Language acquisition in the absence of explicit negative evidence: How important is starting small? *Cognition*, 72(1):67–109, 1999.

Shi, Yangyang, Larson, Martha, and Jonker, Catholijn M. Recurrent neural network language model adaptation with curriculum learning. *Computer Speech & Language*, 33(1):136–154, 2015.

Skinner, Burrhus F. Reinforcement today. *American Psychologist*, 13(3):94, 1958.

Spitkovsky, Valentin I, Alshawi, Hiyan, and Jurafsky, Daniel. Baby steps: How âĂIJless is moreâĂİ in unsupervised dependency parsing. *NIPS: Grammar Induction, Representation of Language and Language Learning*, pp. 1–10, 2009.

Supancic, James S and Ramanan, Deva. Self-paced learning for long-term tracking. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2379–2386, 2013.

Tang, Kevin, Ramanathan, Vignesh, Fei-Fei, Li, and Koller, Daphne. Shifting weights: Adapting object detectors from image to video. In *Advances in Neural Information Processing Systems*, pp. 638–646, 2012a.

Tang, Ye, Yang, Yu-Bin, and Gao, Yang. Self-paced dictionary learning for image classification. In *Proceedings of the 20th ACM international conference on Multimedia*, pp. 833–836. ACM, 2012b.

Zhang, Dingwen, Meng, Deyu, and Han, Junwei. Co-saliency detection via a self-paced multiple-instance learning framework. *IEEE transactions on pattern analysis and machine intelligence*, 39(5):865–878, 2017.