**SAFE WEAKLY SUPERVISED LEARNING USING SGAN**

**INTRODUCTION**

**OVERVIEW OF THE PROJECT**

SAFEW (Safe weakly Supervised Learning) we study weakly supervised learning where a large amount of data supervision is not accessible. This includes incomplete supervision, where only a small subset of labels is given, such as semi-supervised learning and domain adaptation; ii) inexact supervision, where only coarse-grained labels are given, such as multi-instance learning and iii) inaccurate supervision, where the given labels are not always ground-truth, such as label noise learning.

Unlike supervised learning which typically achieves performance improvement with more labeled examples, weakly supervised learning may sometimes even degenerate performance with more weakly supervised data. Such deficiency seriously hinders the deployment of weakly supervised learning to real tasks. It is thus highly desired to study safe weakly supervised learning, which never seriously hurts performance. To this end, we present a generic ensemble learning scheme to derive a safe prediction by integrating multiple weakly supervised learners.

We optimize the worst-case performance gain and lead to a maximin optimization. This brings multiple advantages to safe weakly supervised learning. First, for many commonly used convex loss functions in classification and regression, it is guaranteed to derive a safe prediction under a mild condition. Second, prior knowledge related to the weight of the base weakly supervised learners can be flexibly embedded. Third, it can be globally and efficiently addressed by simple convex quadratic or linear program.

Finally, it is in an intuitive geometric interpretation with the least square loss. Extensive experiments on various weakly supervised learning tasks, including semi-supervised learning, domain adaptation, multi-instance learning and label noise learning demonstrate our effectiveness.

MACHINE learning has achieved great success in numerous tasks, particularly in supervised learning such as classification and regression. But most successful techniques, such as deep learning

• require ground-truth labels to be given for a big training data set. It is noteworthy that in many tasks, however, it can be difficult to attain strong supervision due to the fact that the hand-labeled data sets are time-consuming and expensive to collect. Thus, it is desirable for machine learning techniques to be able to work well with weakly supervised data.

•Compared to the data in traditional supervised learning, weakly supervised data does not have a large amount of precise label information. Weakly supervised data is important in machine learning and commonly appear in many real applications. More specifically, three types of weakly supervised data commonly exist

SAFEW (SAFE Weakly Supervised Learning), which learns the final prediction by integrating multiple weakly supervised learners, and weakly supervised data is more important in machine learning and commonly appear in many real-time applications.

In this study we mainly used for GAN models for safe weakly supervised learning. Generative adversarial networks (GANs) are an exciting recent innovation in machine learning.

GANs are generative models: they create new data instances that resembles your training Datas. For example, GANs can create images that look like photographs of human faces, even though the faces don't belong to any real person. In GAN (generative adversarial network) has two parts: The generator learns to generate plausible data. The generated instances become negative training examples for the discriminator. The discriminator learns to distinguish the generator's fake data from real data. GANs are one kind of generative model. More formally, given a set of data instances X and a set of labels Y: Generative models capture the joint probability p (X, Y), or just p(X) if there are no labels. In GAN model process is

• Random input is given to the generator.

•Generator produces Fake images.

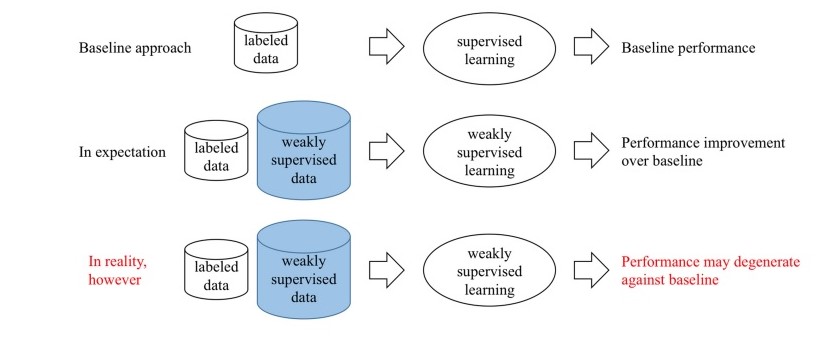
•Input from the fake images and real images is given to the Discriminator.

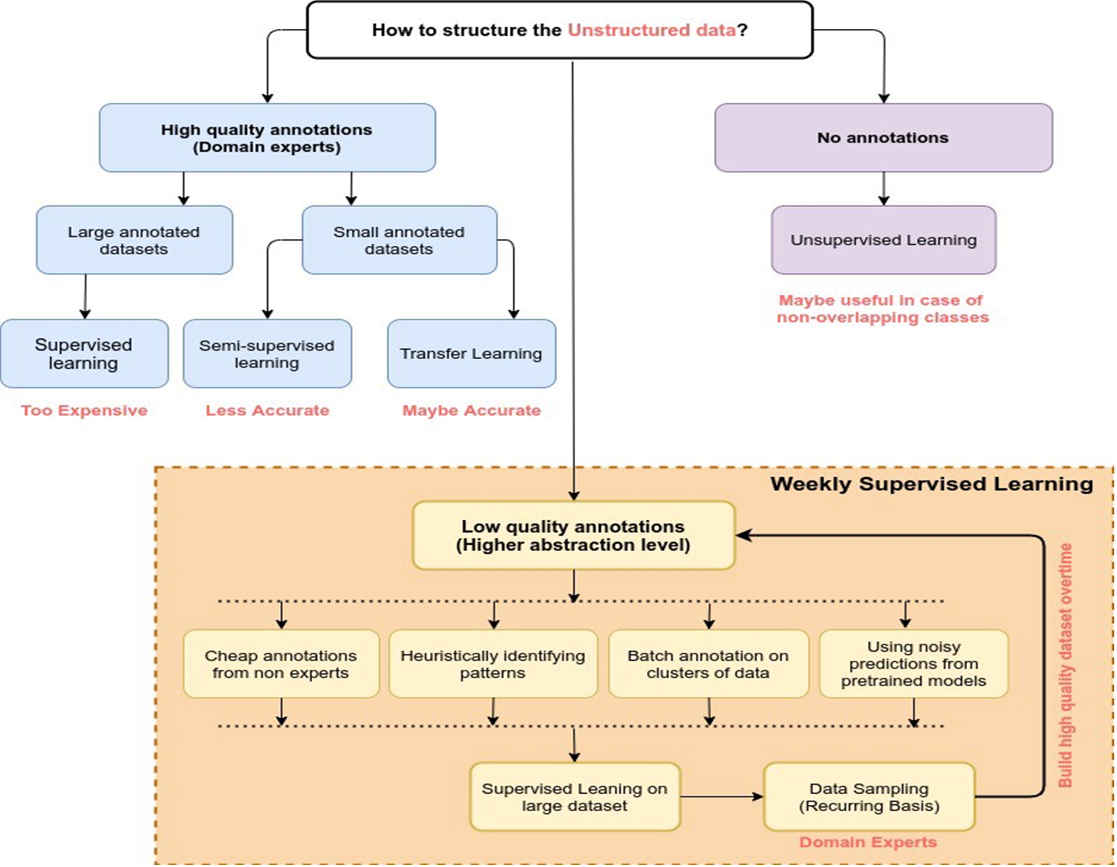
•Discriminator is trained according to how to discriminate between the real and fake images. (Binary classifications).

In this Weakly supervised learning the problem definitions are, Weakly Supervised learning where a large amount of data supervision is not accessible, and it is a general ensemble learning scheme, SAFEW which learns the final prediction by integrating multiple weakly supervised learning. It proposes a maximum and minimum framework, which maximizes the performance gain in the worst case, and it is a generic ensemble learning scheme to derive a safe prediction by integrating multiple weakly supervised learning or learners.

Weakly supervised learning may be not safe, i.e., it may degenerate the performance with the usage of weakly supervised data.

**BASE PAPER ARCHITECTURE**





**LITERATURE SURVEY**

**A BREIF INTRODUCTION TO WEAKLY SUPERVISED LEARNING**

Title: A Brief Introduction to Weakly Supervised Learning  
 Author: Zhi - Hua -Zhou

**PROPOSED SYSTEM**

Supervised learning techniques construct predictive models by learning from a large number of training examples, where each training examples has label indicating its ground- truth output. It is desired for machine learning techniques to work with weak supervision.

Weakly supervised learning, focusing on three typical types of weak supervision: incomplete supervision where only a subset of training data is given with labels; inexact supervision where the training data are given with only coarse – grained labels; inaccurate supervision where the given labels are not ground - truth

**MERITS**

Croud - sourcing is commonly used as a cost - saving way to collect labels for training data.

**DEMERITS**

The available labeled data is noisy or obtained from an imprecise source.

**CLASSIFICATION IN THE PRESENCE OF LABEL NOISE**

Title: Classification in the presence of label Noise  
 Author: Benoit Frenay & Michel Verley Sen.

**PROPOSED SYSTEM**

Label noise is an important issue in classification with many potential negative consequences. Definitions and sources of label noise are considered and a taxonomy of the types of label noise is proposed. The potential consequences of label noise are discussed. Label noise-robust, label noise cleansing, and label noise –tolerant algorithms are reviewed. Label noise consist of mislabeled instances: no additional information is assumed to be available like e.g., Confidences on labels.

**MERITS**

A solution of weakly supervised learning is to make assumption that allows selecting a compromise between naively using instances as they are and seeing any instances as possibly mislabeled.

**DEMERITS**

An identification problem occurs in practical inference mislabeled instances are difficult to distinguish form correctly labeled instances.

**SGAN (Semi \_ Supervised Generative Adversarial Networks)**

The rule of SGAN training are thus follows:

•The global generator and discriminator are training using the local discriminators and generators, respectively, the local network is trained with their fixed local opponent.

•Semi supervised learning is the challenging problem of training a classifier in a dataset that contains a small number of labeled example and a much large number of unlabeled examples.

•The discriminator model can be used as a starting point for developing a classifier model in same cases.

**MODULES**

**• Loading dataset with 70% unlabeled and 30% labeled.**

**•Creating The Generator.**

**•Creating the Discriminator for the feature extraction.**

**•Training the Supervised dataset (labeled).**

**•Testing the unsupervised dataset (unlabeled).**

**LOADING DATASET**

•MNIST (Modified National Institute of Standards and Technology dataset) dataset is used for training data and testing data.

•MNIST set is a large collection of handwritten digits.

•It is a very popular dataset in the field of image processing.

•It is a dataset of 60,000 small square 28\*28-pixel grayscale images of handwritten single digits between 0 and 9.

**CREATING THE GENERATOR**

•The generator part of a GAN learns to create fake data by incorporating feedback form the discriminator.

•It learns to make the discriminator classify its output as real.

•Generator training requires tighter integration between the generator and the discriminator than discriminator training requires.

**CREATING THE DISCRIMINATOR FOR THE FEATURE EXTRACTION**

•The discriminator is a GAN is simply a classifier.

•It tries to distinguish real data from the data created by the generator

•It clouds use any network architecture appropriate to the type of data it’s classifying.

•The discriminator connects to two loss function.

•During discriminator training, the discriminator ignores the generator loss and just uses the discriminator loss.

**Training the supervised dataset(labeled)**

•Training data is the data use to train an algorithm or machine learning model to predict the outcome design the model to predict.

•Test data is used to measure the performance, such as accuracy or efficiency of the algorithm that are using to train the machine.

•These datasets are designed to train or “supervise” algorithms into classifying data or predicting outcomes accurately.

•Training data is also known as training dataset, learning set and training set.

**Testing the unsupervised dataset(unlabeled)**

•Unsupervised learning there is no training dataset and outcomes are unknown.

•Unsupervised learning uses unlabeled training dataset.

•Unlabeled data is data that comes with no tag

**IMPLEMENTATION OF BASE PAPER**

Within Deep Learning, a Convolutional Neural Network or CNN is a type of artificial neural network, which is widely used for image/object recognition and classification. Deep Learning thus recognizes objects in an image by using a CNN. A convolution neural network (CNN) is a type of artificial neural network. It used in image recognition and processing that is specifically designed to process pixel data. There are Three types of layers that make up the CNN

•Convolutional Layers

•Pooling Layers

•Fully Connected Layers

**CONVOLUTIONAL LAYERS**

The CNN and its applications are Convolutional Neural Network is a type of deep learning neural network that is artificial. It is employed in computer vision and image recognition. This procedure includes the following steps: OCR and image recognition. Detecting objects in self-driving cars.

The convolutional Neural Network CNN works by getting an image, designating it some weightage based on the different objects of the image, and then distinguishing them from each other. CNN requires very little pre-process data as compared to other deep learning algorithms.

Convolutional Neural Network (CNN) is a deep learning method and has achieved better results in detecting and segmenting specific objects in images in the last decade than conventional models such as regression, support vector machines or artificial neural networks.

CNN is a supervised type of Deep learning, most preferable used in image recognition and computer vision.CNN can be applied on any 2D and 3D array of data.

**POOLING LAYERS**

The act of sharing or combining two or more things: the pooling of resources. Pooling layers are used to reduce the dimensions of the feature maps. Thus, it reduces the number of parameters to learn and the amount of computation performed in the network. The pooling layer summarizes the features present in a region of the feature map generated by a convolution layer.

In Computer Vision, Deep Learning, Machine Learning. Pooling in convolutional neural networks is a technique for generalizing features extracted by convolutional filters and helping the network recognize features independent of their location in the image.

The pooling operation involves sliding a two-dimensional filter over each channel of feature map and summarizing the features lying within the region covered by the filter.

Pooling Layer The main purpose of pooling layer is to progressively reduce the spatial size of the input image, so that number of computations in the network are reduced. Pooling performs down sampling by reducing the size and sends only the important data to next layers in CNN.

What is the main purpose of pooling layers?

In a convolutional neural network, pooling layers are applied after the convolutional layer. The main purpose of pooling is to reduce the size of feature maps, which in turn makes computation faster because the number of training parameters is reduced.

Pooling is a fixed operation and convolution can be learned. On the other hand, pooling is a cheaper operation than convolution, both in terms of the amount of computation that you need to do and number of parameters that you need to store (no parameters for pooling layer).

**FULLY CONNECTED LAYERS**

Fully connected layer is simply, feed forward neural network. Fully connected layers form the last few layers in the network. The input to the fully connected layer is the output from the pooling or convolutional layer. A fully connected layer multiplies the input by a weight matrix and then adds a bias vector. The convolutional (and down-sampling) layers are followed by one or more fully connected layers. As the name suggests, all neurons in a fully connected layer connect to all the neurons in the previous layer.

Dense layer, also called fully-connected layer, refers to the layer whose inside neurons connect to every neuron in the preceding layer. For the same reason as why two-layer fully connected feedforward neural networks may perform better than single-layer fully connected feedforward neural networks: it increases the capacity of the network, which may help or not.

The difference between convolutional layer and fully connected layers are,

Neural networks are a set of dependent non-linear functions. Each individual function consists of a neuron (or a perceptron). In fully connected layers, the neuron applies a linear transformation to the input vector through a weight's matrix.

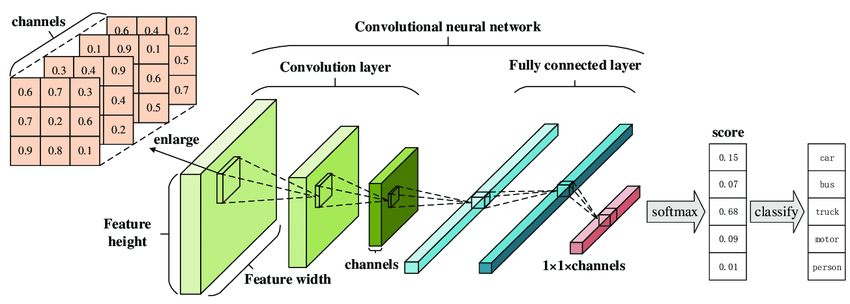
Fully-connected layers, also known as linear layers, connect every input neuron to every output neuron and are commonly used in neural networks. It consists of 128\*3 neurons with three different sizes (3,4 and 5). Then max-pool layer. Output of max-pool layer is concatenated and vector of length 384 is formed which then is inputted to fully connected layer.

In short, each of the 9216 neurons will be connected to all 4096 neurons. That is why the layer is called a dense or a fully-connected layer. As others have said it above, there is no hard rule about why this should be 4096.

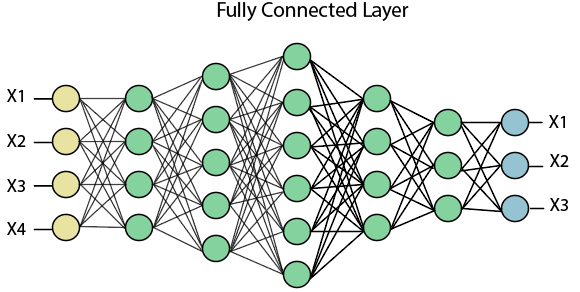
**ARCHITECTURE OF CONVOLUTIONAL NETWORK**



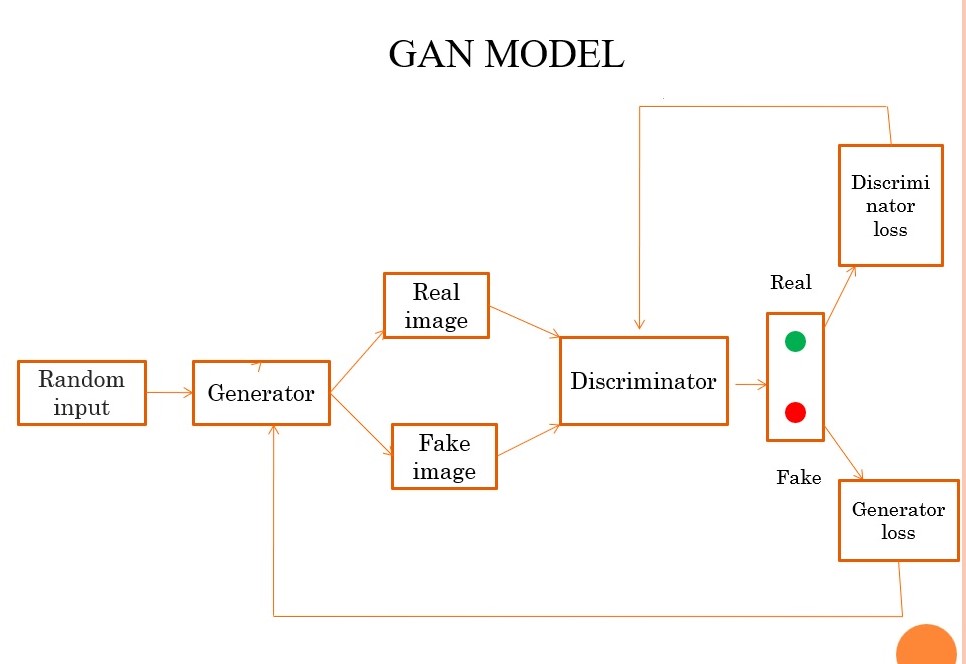
**ARCHITECTURE OF POOLING LAYERS**

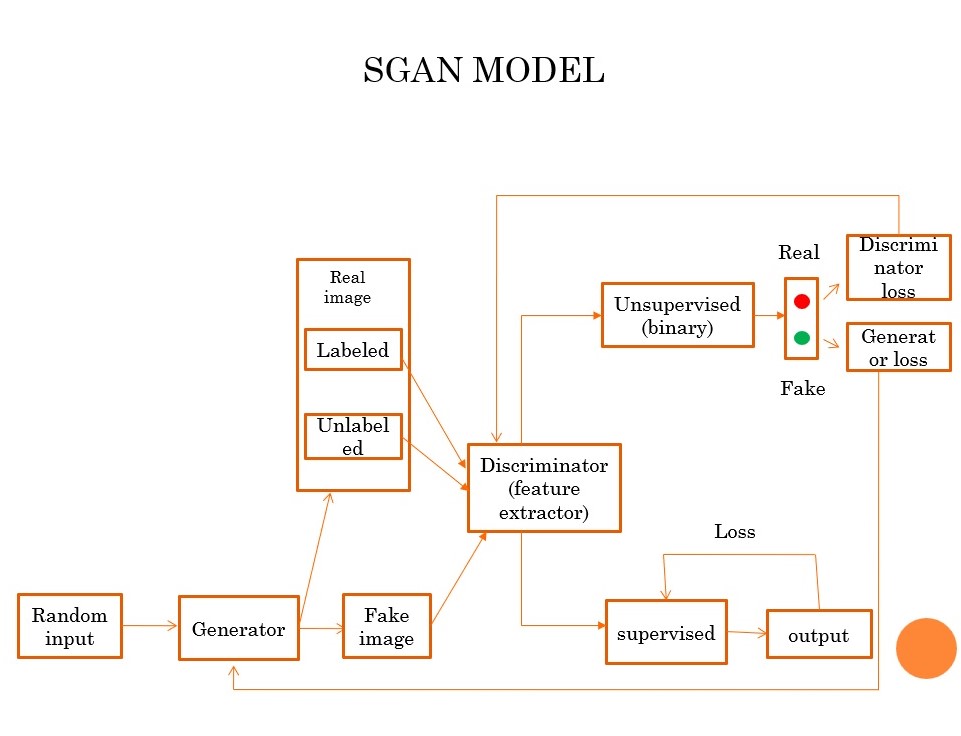


**ARCHITECTURE OF FULLY CONNECTED LAYER**



**ARCHITECTURE OF PROPOSED SYSTEM**

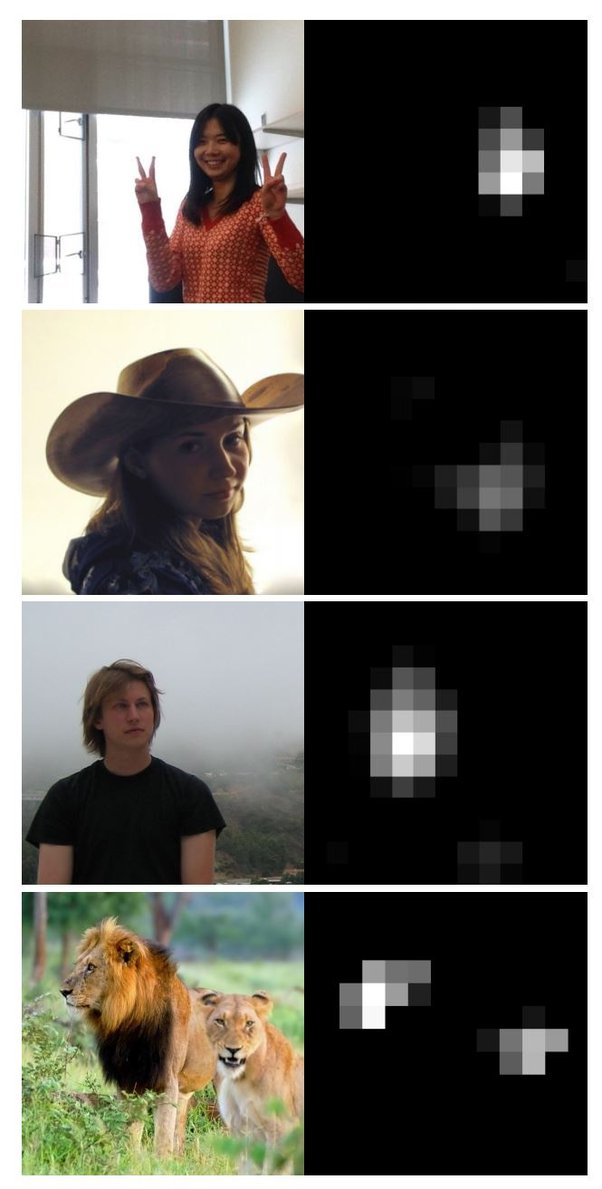




**THE DIFFERNCE BETWEEN CNN’s AUTOENCODERS, GAN’s,VAE’s**

**CNNs**

These stand for convolutional neural networks. This is a special type of neural network that is designed for data with spatial structure. For example, images, which have a natural spatial ordering to it are perfect for CNNs. Convolutional neural networks are composed of many “filters” which convolve, or “slide”, across the data and produce an activation at every slide position. These activations produce a “feature map”, which represents how much the data at that region activated the filter. More specifically, this activation is calculated by taking the dot product of the filter with the image data. As an example, if we have a filter that is trained to recognize faces, these may be the feature maps it outputs:



(This image is from the paper *Understanding Neural Networks Through Deep Visualization,* [*http://yosinski.com/media/papers/Yosinski\_\_2015\_\_ICML\_DL\_\_Understanding\_Neural\_Networks\_Through\_Deep\_Visualization\_\_.pdf*](http://yosinski.com/media/papers/Yosinski__2015__ICML_DL__Understanding_Neural_Networks_Through_Deep_Visualization__.pdf)*,* great paper, definitely recommend reading it)

What is special about convolutional neural networks is that they are spatially invariant, meaning that regardless of where a salient part of the image shows up, it will be detected by the network. This is because the filter weights don’t change at different parts of the image; since the filter is slid across the image, every part of the image is weighted the same.

This spatial invariance property of a CNN is not just applicable to 2d images, but also to 3D video (<http://www.cs.cmu.edu/~rahuls/pub/cvpr2014-deepvideo-rahuls.pdf>), and even 1D timeseries. CNNs have also been considered as a type of pseudo-recurrent neural network, since the filter can slide across timesteps, instead of sections of the data, allowing it to make its decisions based on datapoints in the past (and potentially in the future as well). This technique of looking at past timesteps is used in Deep mind's Wave Net to generate human voices.

**GANs**

GAN stands for Generative Adversarial Networks. These are a type of *generative* model because they learn to copy the data distribution of the data you give it, and therefore can generate novel images that look alike.

The name “adversarial” comes from the two competing networks (adversaries) within GANs that try to outwit each other.

A GAN is often likened to the analogy of a policeman (discriminator) and a counterfeiter (generator). The counterfeiter at first has no idea what real money looks like, so it generates some random looking cash that looks completely fake. Fortunately for the counterfeiter, the policeman has no idea what real money looks like either.

However, we have examples of real cash. So, the police department begins teaching the policeman what real cash looks like, as well as what the discriminator’s fake cash looks like. The policeman can now tell the fake and real cash apart. However, the policeman is super lazy, and just barely learns enough to know the difference between fake and real cash.

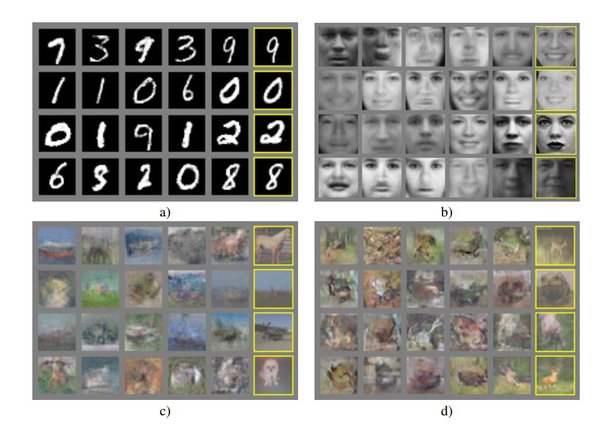
Taking advantage of the policeman’s laziness, the counterfeiter practices even more, and learns to make cash that’s just slightly more realistic, tricking the policeman once again (since he’s really lazy and learned a really bad classifier).

The cycle repeats for some time until (ideally) the policeman cannot tell the difference between the fake cash and the real cash, because the fake cash looks exactly like the real cash. Once this is done, we can just use the generator to make fake cash forever.

Let’s extend this to images. The generator is a neural network that takes in a vector of random variables, Z, and produces an image, I*I*

The discriminator is also a neural network that takes in an image, I*I*, and produces a single output p*p*, deciding the probability that the image is real. When p=1*p=1*, the discriminator believes very strongly that the image is real, and when p=0*p=0*, the discriminator very strongly believes the image is fake.

The discriminator is fed a generator image, which we will denote *igenerator*, and is taught that the image is fake. In more concrete terms, the discriminator maximizes log(1−pgenerator)*log(1−pgenerator)*. The discriminator is then fed a real image, *ireal*, and is taught that the image is real, or, it maximizes *log(preal)*. The generator is trying to do the exact opposite, and is trying to make the discriminator maximize the probability that it believes the fake image is real, so the generator is trying to maximize *log(pgenerator)*. Once we train like this for a while, we will begin to see some pretty realistic photos:



The top two photos are generated photos (except the ones in yellow) of MNIST digits and faces respectively. These perform excellently since numbers and faces both have very convenient and stable structures (they’re always centered, noses are typically at one spot, eyes typically at another). However, once we extend to real-world images, we find that the data distribution of these real-world images is far too complex to model, and we often do not have enough data to produce good generative pictures. It’s still pretty impressive, and you should read the paper: <https://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf>

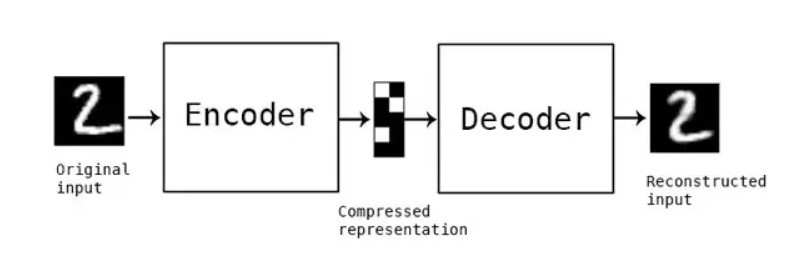
**Autoencoders**

Autoencoders are pretty simple. All they do is take in an input and reproduce the input as closely as possible. If I input a photo of the digit “1”, the autoencoder is supposed to output the exact same photo.

This seems easy and pointless, but some autoencoder configurations can produce interesting results. We typically don’t just have an input and an output layer, because then the network can just copy pixels from the input to the output, which is totally useless. We will usually have a single (or multiple) hidden layer between the input and output layers, that act as bottleneck layers.

Bottlenecks can come in multiple different ways, but I’ll just focus on the easiest one: having fewer hidden neurons.

If the neurons in the hidden layer are less than the number of pixels in the input image, the network has to “compress” the data it sees.



(Image from blog.keras.io).

This compression means that only the most salient features of the image can stay. Whichever features can encode the most information about the data are encoded in the hidden neurons. Essentially, autoencoders with a bottleneck are trying to maximize the information their hidden neurons can store.

This makes autoencoders useful in theory, because if we’re low on supervised training data, we can just feed an autoencoder a bunch of unlabeled data, and it will learn useful features. We can then put those features into a stronger neural network, and have it train on the small supervised dataset. Despite the supervised dataset being small, it would (theoretically) still learn well because it was bootstrapped by the autoencoder.

Unfortunately, autoencoders were not all they were hyped up to be, and this kind of training (called pre-training) is seldom used with autoencoders.

**VAEs**

VAE stands for Variational Autoencoder. A VAE is pretty similar to an autoencoder, but with an interesting twist!

While an autoencoder just has to reproduce its input, a variational autoencoder has to reproduce its output, *while keeping its hidden neurons to a specific distribution*. What this means is that the output of the network will have to get used to the hidden neurons outputting based on a distribution. The consequence of this is that we can generate new images just by sampling from that distribution, and inputting it into the network’s hidden layer.

As an example, let’s pretend our target distribution is a normal distribution with a mean of 0 and a variance of 1. When we input an image into the VAE, the hidden nodes don’t output values that will be directly used by the output, but will instead output mean and variances.

Each of these hidden nodes will act as its own gaussian distribution. We will denote the hidden node values as *hmean* and *hvariance*. We then sample values from an actual normal distribution, which we’ll call z*z*, which is the same size as the hidden layer. We will then multiple z*z* elementwise with *hvariance* and add elementwise with *hmean*. This allows the network to shift the normal distribution and change its variance. This is how it encodes information.

After the elementwise addition and multiplication, we are left with what we call a latent vector. We feed this latent vector into the output layer, and the output layer attempts to produce a copy of the input.

The loss of the autoencoder tries to minimize both the reconstruction loss (how similar the autoencoder’s output was to its input), and its latent loss (how close its hidden nodes were to a normal distribution). The smaller the latent loss, the less information can be encoded, and therefore the reconstruction loss goes up.

As a result, the VAE is locked in a trade-off between the latent loss and the reconstruction loss. If the latent loss is small, our novel generated images will look a lot like the images at train time, but they will both look really bad. If the reconstruction loss is small, then the reconstructed images at train time will look really nice, but our novel generated images will look nothing like the reconstructed images. Obviously, we want both, so it’s important to find a nice equilibrium.

**THE PROPOSED FRAMEWORK**

We first consider a simpler case that the ground-truth label assignment on unlabeled instances is known. Specifically, let f denote the ground-truth label assignment. Remind that our goal is to find a prediction f that maximizes the performance gain against the baseline f 0. One can easily have the objective function as max f2Hu ‘ðf 0;f Þ ‘ðf;f Þ Here ‘ð; Þ refers to a loss function, e.g., the square loss, the hinge loss, etc. Table 2 summarizes some commonly used loss functions for classification and regression.

The smaller the value of the loss function is, the better the performance becomes. However, obviously f is unknown. To alleviate it, inspired by [20], we assume that f is realized as a convex combination of base learners. Specifically, f ¼ Pb i¼1 a if i where a ¼ ½a1; a2;...; ab 0 be the weight of base learners and Pb i¼1 ai ¼ 1. Then we have the following objective instead by replacing the definition of f , max f2Hu ‘ f 0; Xb i¼1 aifi ! ‘ f; Xb i¼1 aifi !: In practice, however, one may still be hard to know about the precise weight of base learners. We further assume that a is from a convex set M to make our proposal more practical, where M captures the prior knowledge about the importance of base learners and we will discuss the setup of M in the later section. Without any further information to locate the weight of base learners, to guarantee the safeness, we aim to optimize the worst-case performance gain, since, intuitively, the algorithm would be robust as long as the good performance is guaranteed in the worst case. Then we can obtain a general formulation for weakly supervised data with respect to classification and regression tasks as, max f2Hu min a2M f 0; X b i¼1 a, if F; X b i¼1 if:

**ANALYSIS**

We in this section show that Eq. (1) has safeness guarantees for the commonly used convex loss functions as listed in Table 2 in the classification and regression tasks of weakly supervised learning. To achieve that, we first introduce a result as follows. Theorem 1. Suppose the ground-truth f can be constructed by base learners, i.e., f 2 ffj Pb i¼1 aifi; a 2 Mg. Let ^f and a^ be the optimal solution to Eq. (1). We have ‘ð^f;f Þ ‘ðf 0;f Þ and ^f has already achieved the maximal performance gain against f 0. Proof. First, we define, Lðf; aÞ ¼ ‘ f 0; Xb i¼1 aifi ! ‘ f; Xb i¼1 aifi !: Since Eq. (1) is a max-min formulation, the following inequality holds for any feasible f and a: Lðf; a^Þ Lð^f; a^Þ Lð^f; aÞ: Let a make f ¼ Pb i¼1 a i fi. By setting f and a to be f 0 and a, we have, ‘ f 0; Xb i¼1 a^ifi ! ‘ f 0; Xb i¼1 a^ifi ! ‘ f 0; Xb i¼1 a i fi ! ‘ ^f; Xb i¼1 a i fi ! Thus, ‘ð^f;f Þ ‘ðf 0;f Þ

**CLASSIFICATION**

Similar to regression tasks, let Cclf be the b b matrix representing the agreement between base learners with elements Cclf ij ¼ E½fiðXÞ >fjðXÞ. Let rclf ¼ ½rclf 1; rclf 2;; rclf b be the vector that represents the agreement between the base learner and the ground-truth, rclf i ¼ E½f ðXÞ >fiðXÞ: Taking classification accuracy as the performance measure, it can be shown that, Theorem 4. The optimal weight a in classification satisfies that rclf ¼ Cclfa. Similarly, we set M as fajC^ clfa 1d; a>1 ¼ 1; a 0g where C^ clf is the unbiased estimation of Cclf , with elements C^clf ij ¼ f> i f j. M is also a convex set. In summary, on one hand, our formulation is able to directly absorb the precise prior knowledge about the importance of learners if available. On the other hand, it is also capable of incorporating with the estimation obtained by covariance matrix analysis on regression and classification tasks when the precise prior knowledge is unavailable.

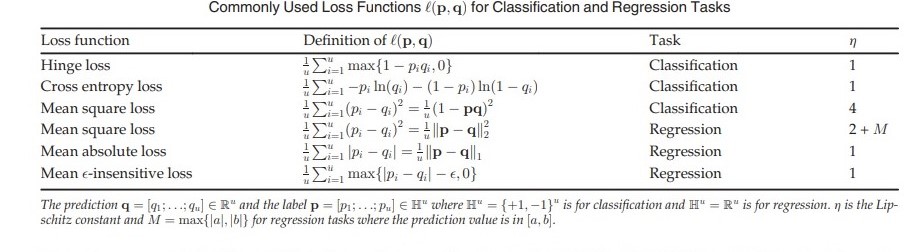
**REGRESSION**

For regression, we have the following theorem, Theorem 5. For regression, suppose ‘ð; Pb i¼1 an if iÞ is convex to a and 8a, and there exists f 2 Ru such that ‘ðf; Pb i¼1 an if iÞ ¼ 0, then Eq. (1) is a convex optimization. We first give a lemma before proving Theorem 5.

Lemma 1. Under the condition in Theorem 5, in optimality, the optimal solution ^f and a^ have the following relation, i.e., ‘ð^f; Pb i¼1 a^ifiÞ ¼ 0. Proof. Assume, to the contrary, ‘ð^f; Pb i¼1 a^ifiÞ 6¼ 0. According to the condition, there exists ~f such that ‘ð~f; Pb i¼1 a^ifiÞ ¼ 0. Obviously, 0 ¼ ‘ð~f; Pb i¼1 a^ifiÞ < ‘ð^f; Pb i¼1 a^ifiÞ. Hence, ^f is not optimal, a contradiction. tu We then prove Theorem 5. Proof. Because of Lemma 1, the form of Eq. (1) for regression task is thus rewritten as, min a2M ‘f 0; X b i¼1 aifi !: Remind that ‘ð; Pb i¼1 A if i is convex to a, therefore, Eq. (1) is a convex optimization. to It is worth noting that the condition in Theorem 5 is rather mild.

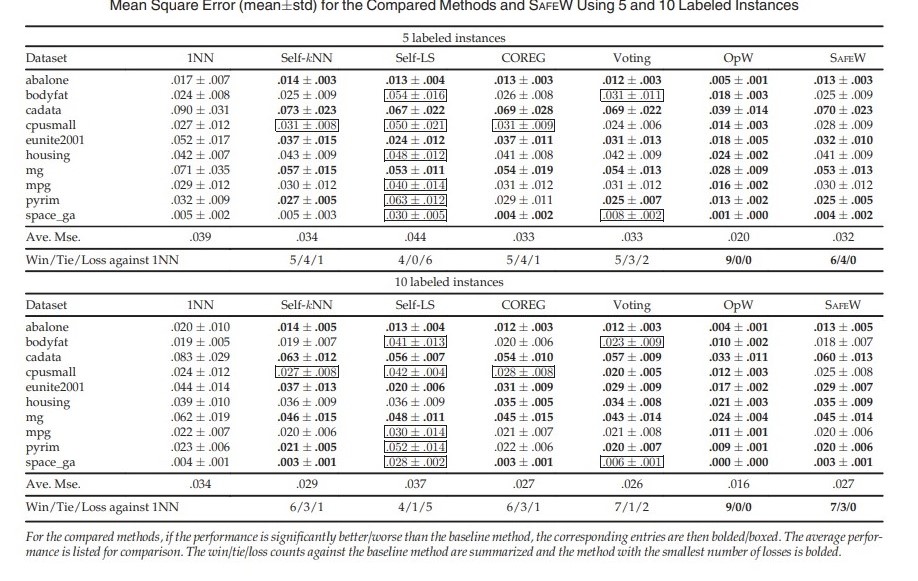
Many regressions loss functions, for example, mean square loss, mean absolute loss [24] and mean -insensitive loss [25], all satisfy such a mild condition in Theorem 5. Depending on Lemma 1 and Theorem 5, the formulation in Eq. (3) can be globally and efficiently addressed for regression. We adopt mean square loss (MSE) as an example to show the optimization procedure since MSE is one of the most popular loss functions for regression. With MSE, Eq. (1) can be written as the following equivalent form which only relates to a. min a2M X b i¼1 A if i f 0, 2: (3) It is evident that Eq. (3) turns out to be a simple convex quadratic program.

Moreover, specifically, by expanding the quadratic form in Eq. (3), it can be rewritten as, min a2M a>Fa v>a; (4) where F 2 Rb b is a linear kernel matrix of Fi's, i.e., Fi ¼ f> i f j and v ¼ ½2f> 1 f 0; 2f> b f 0. Since F is positive semi-definite, Eq. (4) is a convex quadratic program [26] and can be efficiently addressed by off-the shelf optimization packages, such as the MOSEK package.



**Domain Adaptation**

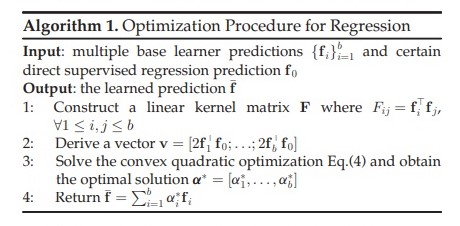
We conduct compared experiments for domain adaptation on two benchmark datasets,4 i.e., 20Newsgrous and Landmine. The 20Newsgroups dataset [50] contains 19,997 documents and is partitioned into 20 different newsgroups. Following the setup in [33], [51], we generate six different cross-domain data sets by utilizing its hierarchical structure. Specifically, the learning task is defined as the top-category binary classification, where our goal is to classify documents into one of the top-categories. For each data set, two top-categories are chosen, one as positive and another as negative. Then we select some subcategories under the positive and negative classes respectively to form a domain. In this work, we use documents from four top-categories: Comp, Rec, Sci and Talk to generate data sets. The Landmine dataset is a detection dataset which contains 29 domains and 9 features. The data from domain 1 to domain 5 are collected from a leafy area; the data from Domain 20 to domain 24 are collected from a sand area. We use the whole data from domain 1 to domain 5 as the source domain and the data from domain 20 to domain 24 as five target domains. For 20newsgroup, following [52], we randomly select 10 percent instances in the target domain as the labeled data and use 300 most important features as the representation. For Landmine, 5 percent instances in the target domain are used as the labeled data. We compare the performance of the proposed SAFEW with the baseline method and 3 state-of-the-art domains.

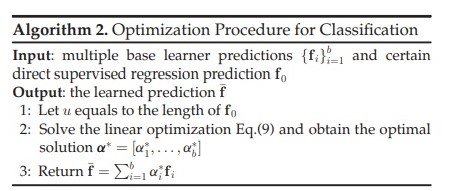


**LABEL NOISE LEARNING**

We conduct experimental comparison for label noise learning on a number of frequently-used classification datasets,6 i.e., Australian, Breast-Cancer, Diabetes, Digit1, Heart, Ionosphere, Splice and USPS. For each data set, 80 percent of instances are used for training and the rest are used for testing. In the training set, 70 percent of instances are randomly selected as the noisy or weakly labeled data and the rest ones are high-quality labeled data. For the noisy labeled data, their labels are randomly reversed with a probability p% where p range from 10 percent to 40 percent with an interval 10 percent. We compare the performance of the proposed SAFEW with the following methods. a) Baseline Sup-SVM method, which is a supervised SVM trained on only high-quality labeled data. b) Bagging, which is regarded as to be robust with label noisy [7]. c) RLR (Robust Logistic Regression) [61], that enhances the logistic regression model to handle label noise. d) 3 classic classification methods: SVM, LR (Logistic Regression), k-NN with regardless of label noise. For LR, the glmfit function in Matlab is used. For k-NN method, k is set to 3. For Sup-SVM and SVM method, Libs package [62] is adopted and the kernel is set to RBF kernel. For Bagging method, we adopt the decision tree as the base learner. For RLR method, the parameter is set to the recommended one. For SAFEW, LR, SVM, and k-NN are invoked as base learners and parameter d is set by 5-fold cross validation from the range ½0:5u; 0:7u. Experiments are repeated for 30 times, and the average classification accuracy is reported.

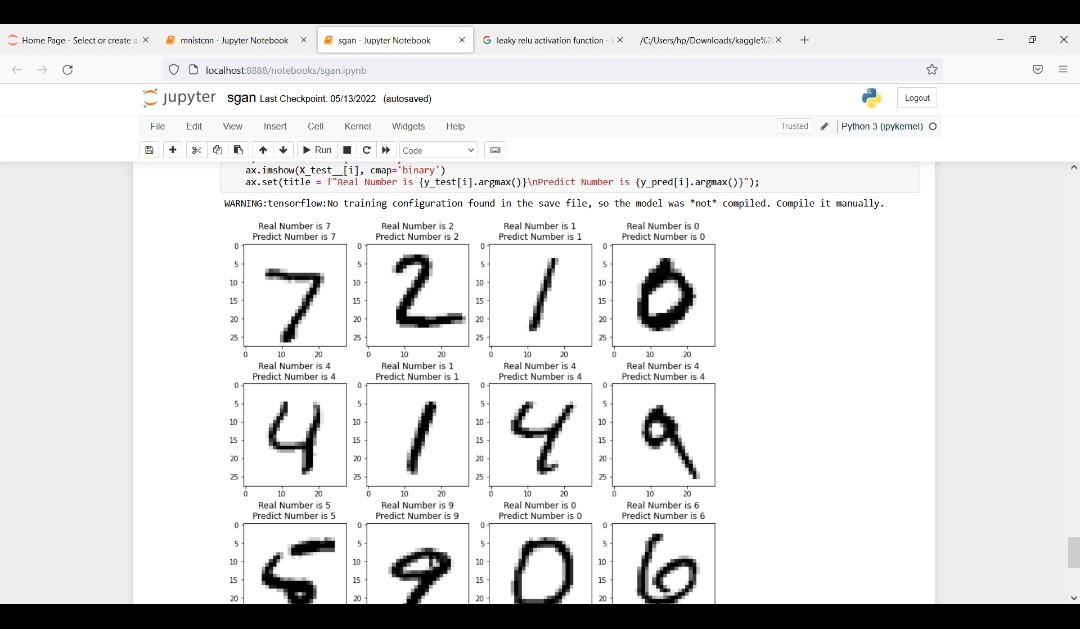
**ALGORITHM**



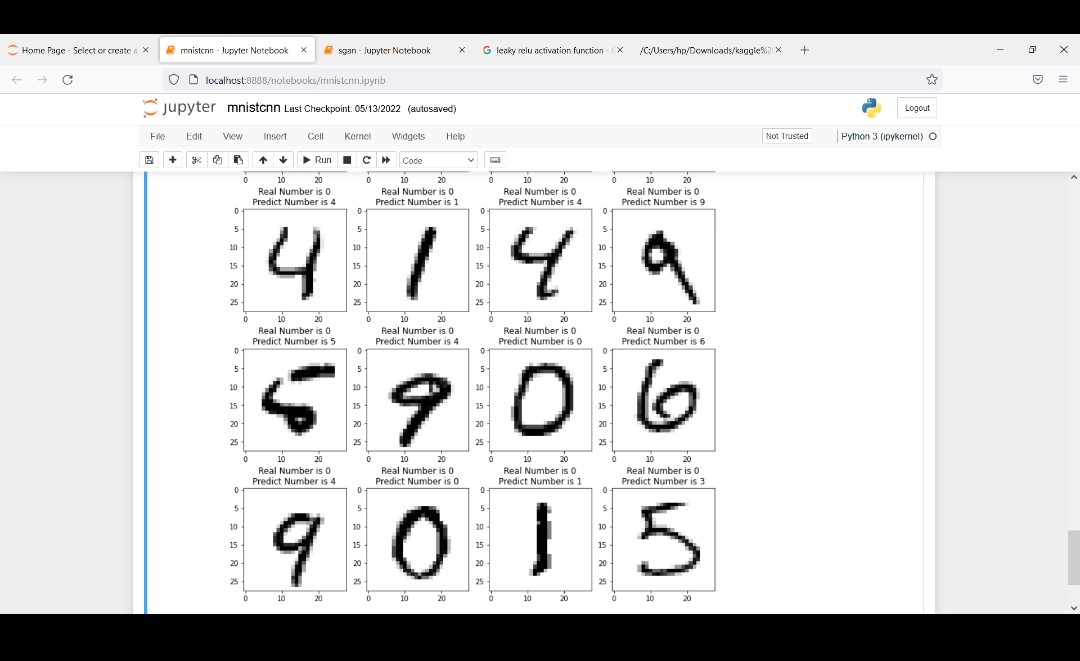


**RESULTS**

**SGAN OUTPUT SCREENSHOT**



**CNN OUTPUT SCREENSHOT**



**COMPARISON OF RESULT**

**Existing system**

•It takes more time for consumption.

•Low average performance.

•Low quality labeled data.

**Proposed system**

•It takes less time for consumption.

•Best average performance.

•High quality labeled data.

**CONCLUSION AND FUTURE ENHANCEMENT**

•Safe weakly supervised learning that will not affect the performance with the use of weakly supervised data.

•A scheme to derive a safe prediction by integrating multiple weakly supervised learners.

•The resultant formulation has a safeness guarantee for many commonly used convex loss functions in classification and regression.

•In future, it is necessary to study safe weakly supervised learning with adversarial example.

**REFERENCE**

**Yu-Feng Li** received the BSc and PhD degrees in computer science from Nanjing University, China, in 2006 and 2013, respectively. He joined the National Key Laboratory for Novel Software Technology at Nanjing University in 2013, and is currently an associate professor. He is a member of the LAMDA group. His research interests include mainly in machine learning. Particularly, he is interested in weakly supervised learning, statistical learning and optimization. He has received outstanding doctoral dissertation award from China Computer Federation (CCF), outstanding doctoral dissertation award from Jiangsu Province and Microsoft Fellowship Award. He has published more than 40 papers in top-tier journals and conferences such as the Journal of Machine Learning Research, the IEEE Transactions on Pattern Analysis and Machine Intelligence, the Artificial Intelligence, the IEEE Transactions on Knowledge and Data Engineering, ICML, NIPS, AAAI, etc. He is/was served as an editorial board member of machine learning journal special issues, co-chair of ACML18 workshop and ACML19 tutorial, and a senior PC member of top-tier conferences such as IJCAI19/17/15, AAAI19.

**Zhi-Hua Zhou** (S’00-M’01-SM’06-F’13) received the BSc, MSc, and PhD degrees in computer science from Nanjing University, China, in 1996, 1998 and 2000, respectively, all with the highest honors. He joined the Department of Computer Science & Technology, Nanjing University as an assistant professor in 2001, and is currently professor, head of the Department of Computer Science and Technology, and dean of the School of Artificial Intelligence; he is also the founding director of the LAMDA group. His research interests include in artificial intelligence, machine learning and data mining.

He has authored the books Ensemble Methods: Foundations and Algorithms and Machine Learning (in Chinese), and published more than 150 papers in top-tier international journals or conference proceedings. He has received various awards/honors including the National Natural Science Award of China, the IEEE Computer Society Edward J. McCluskey Technical Achievement Award, the PAKDD Distinguished Contribution Award, the IEEE ICDM Outstanding Service Award, the Microsoft Professorship Award, etc. He also holds 24 patents. He is the editor-in-chief of the Frontiers of Computer Science, associate editor-in-chief of the Science China Information Sciences, Action or associate editor of the Machine Learning, IEEE Transactions on Pattern Analysis and Machine Intelligence, ACM Transactions on Knowledge Discovery from Data, etc.

He served as associate editor-in-chief for Chinese Science Bulletin (2008-2014), associate editor for IEEE Transactions on Knowledge and Data Engineering (2008-2012), IEEE Transactions on Neural Networks and Learning Systems (2014-2017), ACM Transactions on Intelligent Systems and Technology (2009-2017), Neural Networks (2014-2016), etc.

He founded ACML (Asian Conference on Machine Learning), served as Advisory Committee member for IJCAI (2015-2016), Steering Committee member for ICDM, PAKDD and PRICAI, and chair of various conferences such as general co-chair of ICDM 2016 and PAKDD 2014, Program co-chair of AAAI 2019 and SDM 2013, and area chair of NIPS, ICML, AAAI, IJCAI, KDD, etc. He is/was the chair of the IEEE CIS Data Mining Technical Committee (2015-2016), the chair of the CCF-AI (2012- ), and the chair of the CAAI Machine Learning Technical Committee (2006-2015). He is a foreign member of the Academy of Europe, and a fellow of the ACM, AAAI, AAAS, IEEE, IAPR, IET/IEE, CCF, and CAAI. He is a fellow of the IEEE

**Lan-Zhe Guo** received the BSc degree in 2017. He is currently working toward the PhD degree in the National Key Laboratory for Novel Software Technology at Nanjing University, China. His research interests include in machine learning. Particularly, he is interested in weakly-supervised learning