Fighting Label Noise: Empirical Study of Robust Classification Methods

Abstract

Most classification training methods assume that the ground truth labels are fully trustworthy. However, the datasets are usually accumulated using a combination of human hand-labeling and, sometimes, automatic labeling techniques. Those techniques can very likely introduce some flipping noise into the labels, which is then inevitably inherited by the model's predictions, if trained naively. We investigate on two label noise robust training methods: backward correction [1] and co-teaching [2] and their performance on FashionMNIST [3] and CIFAR [4] datasets with artificially corrupted labels. Additionally, in this work we test a method for finding transition matrices in corrupted datasets, which is based on anchor points [5]. This work is organized as follows: in introduction, we state the label noise problem in detail, in related work we discuss existing methods that tackle the problem, in the methods section we describe the algorithms that are tested in this work [1, 3, 2], in the experiments we describe the exact modus operandi of estimating the methods as well as the obtained results and discussion, and in conclusion we discuss the results and speculate on future work.

1 Introduction

One of the tasks in the field of Machine Learning is to leverage the existing data in a semi brute-force way to learn some laws about the phenomena in nature and culture. One of the most known subproblems in machine learning is called *classification* — a process of determining a discrete class that a given object with given features belongs in. Classification is a well-studied problem, with classic modeling methods such as SVMs [6], gradient boosting [7], as well as recent advances for representing more complex data objects such as natural language phrases [8] and images [9].

One thing classification training methods rely on is trustworthiness of the ground truth that is used to calculate the error and update the model. It is harder to validate that no error was introduced at any point of the pipeline of creating the datasets, especially as their sizes grow recently. There's a trade-off between quality of the data labels and the cost of labeling. Some domains are more subjective to human eyes then the other, or require expert labelers which is an incredible cost increase.

As en example, Imagenet [10] is wildly used to pretrain large CNN models which are then finetuned on smaller datasets. However, it is hard to verify the corretness of the labels, it is believed somewhere around 6% of them are incorrect. Therefore, a problem of increasing robustness to that noise exists and it is important.

There are three main types of label noise:

- **Random classification noise**. The distribution of the label noise is independent of the true label of the object or its features
- **Class-dependent label noise**. The distribution of the label noise depends on the true label of the object
- **Instance-dependent label noise**. The distribution of the label noise depends both on the true label and the features of the object

The problem that we tackle in this work is the second one, the problem of **classification robust to class-dependent label noise**.

An example of a real-life application of this problem would be as follows: we have a dataset, there's some confusion in the labels: for example, class 2 is often labeled as 1, class 1 is consistent, while class 3 is often labeled as both classes 2 and 1. If the training method is robust to the noise, it will account for that confusion and adjust the probabilities of the model, so that, eventually, while observing the test scenes, the confusion between those classes is reduced.

Backward correction [1] (also known as unbiased estimator) is the first method that we use. Given a dataset with noisy labels, the method leverages known noise flip rates between the classes and uses them to transform the loss function, making it equivalent to one with denoised ground truth labels.

Co-teaching [2] is another method that we use in this work. The method uses the information about the noise rate in the dataset and trains the networks only on the samples with smallest loss value. Additionally, it uses two networks which act like peer-reviewers, showing each other the "safe" samples the target function can be learnt from.

Since both of those training methods require knowing the flip rates (transition matrix) in one form or another, this work also uses anchor points method [5] to estimate them. Let's use binary classification as an example. The method leverages the fact that, after the training, the network will have seen at least two samples with the same distribution of features, but with different labels (-1 and +1). So, a sample from the training set with the highest probability of it being in the positive class reveals to us the information about the transfer matrix.

2 Related work

The problem of label noise is closely related to the problem of domain adaptation. It can arguably be stated that the problem of adapting to label noise is a case of domain adaptation: we have a training distribution available and modified testing distribution, that is different (even if related). For example, "Classification with Noisy Labels by Importance Reweighting" [11] is widely based on the same idea used for domain adaptation [12] and importance reweighting [13]. In particular, in [13], a transformation of the empirical risk function is introduced, so that we can optimize the function for one domain through another domain. A constraint for this method is that the support of the training domain distribution should contain the support of target domain distribution. So, to summarize, importance reweighting for noisy label classification [11] is a fast simple technique of modifying the training method, however a disadvantage is that it eventually requires known flip rates, and it's only formulated for binary classification. The method also relies on the estimation of the conditional distribution of the noisy samples.

Method of Unbiased estimators described in [14] is a simpler method of modifying the loss using a parameter that can be tuned by cross-validation. The presence of this parameter complicates the training, which can be considered a disadvantage of the method. And, of course, again, the method requires known noise rates for a binary setting.

3 Methods

3.1 Backward correction [1]

This method extends the method of unbiased estimators [14] to a multi-class setting. The architecture of the model that is optimized with this training method is not important, since the method is architecture and domain agnostic. The main inside of the method is the modification of the loss function using the noise transition matrix, therefore the model architecture and optimization method are only described in the experiments section.

3.1.1 Problem statement

Assume we have a feature space $\mathcal{X} \in \mathbb{R}^d$ and label space $\mathcal{Y} = \{e^i : i \in [c]\}$, where $[c] = \{1, ..., c\}$, c is a positive integer. The task is to learn the unknown distribution $p(x, y) = p(y|x)p(x)$ over the Cartesian product of the features and labels $\mathcal{X} \times \mathcal{Y}$.

The problem is, however, that each label in the set y is flipped with probability $p(\hat{y}|y)$, so, for class i the probability to be flipped to j is denoted as $p(\hat{y} = e^j|y = e^i) = T_{ij}$. Let's denote the full matrix of the flipping probabilities as $T \in \mathbb{R}^{c \times c}$.

So, to summarize, the problem is to estimate the unknown distribution $p(x, y) = p(y|x)p(x)$ from the samples of its modified (corrupted) version $p(x, \hat{y}) = \sum_y p(\hat{y}|y)p(y|x)p(x)$.

3.1.2 Corrected loss function

In this setting, the loss function used in the training method is denoted as follows (using a per-class notation):

$l(p(y|x)) = (-\log p(y = e^1|x), -\log p(y = e^2|x), ..., -\log p(y = e^C|x))^T \in \mathbb{R}^c$

Theorem 1 from [1] states, having known matrix T, a minimizer of expected risk of $l(p(y|x))$ over $p(x, y)$ is the same as a minimizer of expected risk of $l^{\leftarrow}(p(y|x)) = T^{-1}l(p(y|x))$ over $p(x, \hat{y})$. Or, more formally:

$$\arg\min_{p(y|x)} \mathbb{E}_{x,\hat{y}}[l^{\leftarrow}(y, p(y|x))] = \arg\min_{p(y|x)} \mathbb{E}_{x,y}[l(y, p(y|x))] \tag{1}$$

In other words, we build an unbiased estimator of the loss function, correcting the expectation under the distribution of noisy labels to be equal to the expectation under the distribution of clean labels.

This is a multi-class generalization of Theorem 3 from [14]. A more general definition of the theorem can be found in [15], Theorem 3.2.

3.2 Co-teaching [2]

This method is based on trying to only learn on clean samples using a second network as a kind of a peer reviewer. This is a training method and the only limitation for the model's architecture is that the architectures of both of the models should be the same. As for optimization, any gradient descent-like optimizer can work with this method i.e. Adam [16].

3.2.1 Problem statement

The problem statement here is almost the same to that explained in 3.1.1. However, the assumption of knowing transition matrix T is relaxed here down to knowing the noise rate $\epsilon = \mathbb{E}[p(\hat{y} \neq y)]$, which is just the approximate fraction of the samples in the dataset that have noisy labels. Having known T, the parameter can be calculated. Without known T, it can be tuned (it's relatively easy, since the value is from 0 to 1).

3.2.2 Co-teaching method

Let $\mathcal{D}$ denote the training set and $\hat{\mathcal{D}}$ denote the mini-batch. Our aim is to train two estimators: f and g. Let $R(e)$ be some value between 0 and 1, which is modified with each epoch e. At epoch e, for each batch, we calculate the two loss functions: $l(f, \hat{\mathcal{D}})$ and $l(h, \hat{\mathcal{D}})$. Let $\hat{\mathcal{D}}_f$ be top $R(e)\%$ small-loss instances on f from $\hat{\mathcal{D}}$. By analogy, $\hat{\mathcal{D}}_g$ is the top $R(e)\%$ small-loss instances on g from $\hat{\mathcal{D}}$.

For each batch, the loss for each network is cross-masked, meaning the most certain samples of one network are backpropagated through the other network:

$$\begin{aligned} l(f, \hat{\mathcal{D}}) &= l(f, \hat{\mathcal{D}}_g) \\ l(g, \hat{\mathcal{D}}) &= l(g, \hat{\mathcal{D}}_f) \end{aligned} \tag{2}$$

The algorithm runs for $e_{\max}$ epochs. For each epoch e the least-loss samples percentage is calculated as follows: it linearly decreases from 1 to ϵ until some threshold epoch e_{t}. After that epoch, $R = \epsilon$ until the rest of the training. Or, mathematically:

$$R(e) = 1 - \min(\frac{e}{e_{\text{t}}}\epsilon, \epsilon) \tag{3}$$

After the final epoch, we can pick any of the two networks for testing and inference.

Eventually, the algorithm utilizes the following two things to handle the label noise:

- Exploiting the fact that the networks learn easy patterns first [17] gives us hope that, during the first epochs, the networks are learning to reduce their loss on the instances that don't have any noise in their labels. However, at the later epochs, they will eventually start overfitting, and that's why R is reduced as a function of the current epoch.
- To increase robustness and add an additional layer of outlier-robustness to the training, a second network is added to the pipeline, which is largely based on the idea of co-training [18]. Even though the two networks architectures are identical, their weights are initialized in a different fashion (i.e. Xavier initialization [19]), and they might learn different patterns in different order. To boost their joint convergence and protect them from outliers, they can reveal to each other the samples that they've learnt so far, which have a high probability of being the clean ones.

3.3 Anchor points method

The method of anchor points is based on ideas from [5] and, in multi-class case, from [1]. Again, the method mostly consists of instructions on how to train and inference the network in a correct way to produce the matrix T, so the architecture and the optimization algorithm can be of user's choice.

3.3.1 Problem statement

Assume we have a feature space $\mathcal{X} \in \mathbb{R}^d$ and label space $\mathcal{Y} = \{e^i : i \in [c]\}$, where $[c] = \{1, ..., c\}$, c is a positive integer. Each label in the set y is flipped with probability $p(\hat{y}|y)$, so, for class i the probability to be flipped to j is denoted as $p(\hat{y} = e^j | y = e^i) = T_{ij}$. Let's denote the full matrix of the flipping probabilities as $T \in \mathbb{R}^{c \times c}$. The task is to find T from $p(\hat{y}|x)$.

3.3.2 Method description

The method is essentially based on Theorem 3 from [1] which itself is a multi-class extension of anchor points method presented in [11] and partially in [20]. The theorem assumes the following two things about $p(x, y)$:

- $p(x, y)$ contains "perfect examples" of each class, such that for each class c there exists x_c such that $p(y = c|x_c) = 1$
- The model accurately approximates $p(\hat{y}|x)$

Then, for every i and j in the set of all classes $[c]$, the following is true:

$$T_{ij} = p(\hat{y} = e^j | x_i) \tag{4}$$

In other words, this theorem states, that if our model was trained on the clean version of the dataset, it would be absolutely certain about those examples predictions. But, since it was trained on the noisy version of the dataset, its certainty about the class is instead governed by the probability of the label staying clean under the noise.

Thus, the algorithm of finding the matrix would look as follows:

- Train estimator h on the training dataset $\mathcal{D} \sim p(x, \hat{y})$
- Sample D', a portion of $\mathcal{D}$
- For every class i find the "perfect example" $x^i = \arg\max_{x \in D'} p(\hat{y} = e^i | x)$
- Fill the i-th row of the matrix $T_i = [p(\hat{y} = 0|x^i), p(\hat{y} = 1|x^i)], ..., p(\hat{y} = c|x^i)]$

The larger D' is, the closer we are approximating to the true value of T, which is justified by the first out of the two assumptions described above. The second one is harder to justify: though, in practice, a good validation accuracy can indicate that we fit the network good enough to predict the noisy distribution.

4 Experiments

4.1 Datasets

The datasets

- **FashionMNIST5** - 18000 training/validation samples and 3000 images sized (28×28). Each image belongs to one of the three classes: 0 is shirts, 1 is pants and 2 is dress. The transition matrix T is known and equals $\begin{bmatrix} 0.5 & 0.2 & 0.3 \\ 0.3 & 0.5 & 0.2 \\ 0.2 & 0.3 & 0.5 \end{bmatrix}$.
- **FashionMNIST6** - 18000 training/validation samples and 3000 images sized (28×28). Each image belongs to one of the three classes: 0 is shirts, 1 is pants and 2 is dress. The transition matrix T is known and equals $\begin{bmatrix} 0.4 & 0.3 & 0.3 \\ 0.3 & 0.4 & 0.3 \\ 0.3 & 0.3 & 0.4 \end{bmatrix}$.
- **CIFAR** - 30000 training/validation samples and 3000 images sized $(32 \times 32 \times 3)$. Each image belongs to one of the three classes: 0 is airplane, 1 is car and 2 is cat. The transition matrix T is unknown and is estimated using the anchor points method. The experiment is described below.

All images in all datasets are rescaled to values from 0 to 1 before usage for numerical stability of the optimization algorithms.

4.2 Metrics

To evaluate the classification methods, a standard set of metrics was used: **precision**, **recall**, **f1 score** and **top 1 accuracy**. **Recall** is responsible for the ability of the model to label the correct objects, while **precision** is responsible for the amount (or percentage) of correct objects that are labeled positive. **f1 score** is a convenient metric that combines precision and recall into a single one. **Top n accuracy** is similar to classic accuracy, but it counts for class appearing in top n predictions of the model ranked by the probability. All the metrics are calculated separately for each class and then averaged.

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{5}$$

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{6}$$

$$\text{f1} = 2 \times \frac{\text{recall} \times \text{precision}}{\text{recall} + \text{precision}} \tag{7}$$

$$\text{top1accuracy} = \frac{\text{TP} + \text{TN}}{\text{N}} \tag{8}$$

Where TP - true positive answers of the model, TN - true negative answers of the model, FP - false positive answers of the model, FN - false negative answers of the model.

4.3 Noise estimation (anchor points)

4.3.1 Experiment setup

The algorithm used is a Multi-Layer Perceptron (MLP) classifier which is a neural network algorithm suited for image classification. The neural network algorithm is designed with a two-tiered

hidden layer structure that hosts 128 neurons in the first layer and 64 in the second. This configuration is activated by the ReLU function and optimized through the Adam solver, balancing computational efficiency with convergence quality.

This MLP classifier is further augmented with the Calibrated Classifier CV, this method enhances the model's probabilistic predictions to be more representative of the true class probabilities. Which is especially needed when working with noisy labels, as it ensures the predicted probabilities are more accurate. The calibration is done using a sigmoid method within a cross-validation scheme to prevent the model from being biased by its own decisions during training, thereby improving the generalization of the probability estimates on unseen data.

4.3.2 Results

The FashionMNIST0.5 dataset has an average accuracy of 48.2%, with low variability in performance metrics, indicating consistent model behavior across runs. The F1 score's closeness to accuracy suggests balanced precision and recall. The model's estimated transition matrix aligns well with the known matrix, implying effective noise management.

For FashionMNIST0.6, there's a drop in accuracy to 38.9%, with standard deviations similar to the 0.5 dataset, yet the F1 score's slight decrease might imply a precision-recall imbalance, potentially due to increased label noise and its impact on model bias.

CIFAR's average accuracy is lower at 35.9%, with a noticeable F1 score variability pointing to less stable precision-recall balance across runs, reflective of the dataset's greater complexity.

As a self-reflection, the results highlight the challenge of handling label noise in image classification. Specifically, as the noise level increases, as seen in FashionMNIST0.6 dataset, or as the image complexity goes up, as in CIFAR, the accuracy of the model decreases. This indicates the need for more advanced or noise-robust algorithms. The consistency in performance metrics across runs suggests that the evaluation method is stable, yet there is room for improvement in model accuracy. The transition matrix was also a valuable tool, but its effectiveness depends on accurately capturing the true noise distribution, which remains a complex task especially when such distribution is unknown.

As a result, the matrix found for CIFAR dataset $T = \begin{bmatrix} 0.364 & 0.329 & 0.3078 \\ 0.323 & 0.354 & 0.323 \\ 0.314 & 0.321 & 0.364 \end{bmatrix}$.

4.4 Backward correction [1]

4.4.1 Experiment setup

This method requires known flip rates in a form of matrix T.

For FashionMNIST datasets, we chose a fully-connected network with two hidden layers sized 128 and 50% dropout applied after each layer. For the CIFAR dataset, we chose LeNet [21] with two convolutional layers and three linear layers, since, comparing to FashionMNIST, CIFAR is a multi-channel dataset.

For this method, we trained the networks for 40 epochs in batches sized 128, used Adagrad optimizer with learning rate 10^{-2} and weight decay 10^{-6}. We split the training dataset into training and validation $80\%/20\%$. In those experiments, we provide performance averaged over 10 models trained on different samples of the training dataset. We computed two versions of metrics: one for backward correction and one for naive training (which is equivalent to backward correction with $T = I$).

4.4.2 Results

The results are presented in Table 1 (naive and backward correction). We see that every single metric for every single dataset is improved using the backward correction method. This is due to the fact that we correct the loss and account for the noise in the training labels. Although, the quality for naively training and ignoring the noise is quite impressive: one of the reasons might be the fact that we use dropout in our model, as well as scheduled learning rate, which improves generalization

		Metrics			
Model	Dataset	*accuracy@1*	*recall*	*precision*	*f1*
Naive	F5	0.9208 ± 0.0246	0.8812 ± 0.0369	0.8874 ± 0.0333	0.8815 ± 0.0373
	F6	0.875 ± 0.011	0.8124 ± 0.0165	0.8228 ± 0.019	0.8135 ± 0.017
	CIFAR	0.6136 ± 0.0108	0.4204 ± 0.0161	0.4223 ± 0.0156	0.4157 ± 0.0169
Backward Correction	F5	0.933 ± 0.0035	0.8995 ± 0.0052	0.9001 ± 0.0052	0.8996 ± 0.0053
	F6	0.8933 ± 0.0078	0.84 ± 0.0117	0.8467 ± 0.0095	0.8398 ± 0.0123
	CIFAR	$\mathbf{0.7183 \pm 0.0612}$	0.5774 ± 0.0919	0.5626 ± 0.1564	$\mathbf{0.5524 \pm 0.1372}$
Co-teaching	F5	$\mathbf{0.9543 \pm 0.0023}$	0.9314 ± 0.0035	0.932 ± 0.0035	$\mathbf{0.9316 \pm 0.0035}$
	F6	$\mathbf{0.9014 \pm 0.0123}$	0.8521 ± 0.0184	0.8546 ± 0.0159	$\mathbf{0.8506 \pm 0.0196}$
	CIFAR	0.6948 ± 0.0109	0.5422 ± 0.0163	0.5454 ± 0.0175	0.5454 ± 0.0175

Table 1: Result corrected classification metrics. F5 is short for FashionMNIST5, F6 - for FashionMNIST6. All the values are averaged across 10 runs with different train/validation sampling. The format is $\mathrm{mean\_val} \pm \mathrm{std\_val}$

and allows the model to learn even through the noise. For FashionMNIST5 and FashionMNIST6 datasets, the matrix T is known, hence we achieved higher eventual quality. For CIFAR dataset, we used the matrix T that we found using the anchor points method (experiment described above), which only approximates the matrix, therefore that's why the quality might be lower.

Also, the overall lower quality of CIFAR can be explained by the fact that it's a harder dataset: 3 channels (colored images), much more diverse classes. It is probably possible to finetune it better, but the effect of switching from naive training to backward correction is likely to stay the same.

4.5 Co-teaching [2]

This method doesn't require known flip rates, but a vague idea of the noise rate in the dataset can be utilized through the parameter ϵ. The value of ϵ that we used was calculated from T: $\epsilon = 1 - \frac{\sum_{i=0}^{n} diag(T)}{n}$, which is essentially the probability to get an incorrect label in the training dataset, averaged by class.

For this experiment the setup is really similar to backward correction, we use the same architectures and the same batch size, as well as same validation scheme. But this method required us to train for 200 epochs with learning rate 5×10^{-4} using Adam optimizer. We compare the method against the naive training from previous experiment as well as against the backward correction method.

4.5.1 Results

The results are presented in Table 1, where we compare co-teaching with backward correction and naive method. Surprisingly, even though not fully and explicitly using the the noise transition matrices, *co-teaching performs the best for FashionMNIST5 and FashionMNIST6*, but it doesn't outperform backward correction for CIFAR. The reason for that might be as follows: since we only approximate the T matrix for CIFAR, the noise rate ϵ very well might be slightly off. However, an advantage of this method is that it's tuneable: a bunch of different ϵ values can be experimentally tried to increase the accuracy on CIFAR. Also, due to the way co-teaching works, the two networks are much less likely to discriminate noise samples in such a diverse dataset as CIFAR. If in FashionMNIST5 and FashionMNIST6 the features of images are more or less similar from image to image, CIFAR is much complicated and might require additional tuning and augmentation.

5 Conclusion

5.1 Results and personal reflection

To summarize, we implemented methods that adapt to noise in training labels. We compared two methods, one of which uses transition matrices T, against naive training. We did it for three datasets, for one of which we found a transition matrix T using anchor points algorithm. Figure 1 and Figure 2 show the eventual results. On average between the dataset, each single metric is improved by the training correction methods. Co-teaching is the best one on average, with backward correction being a close second. Personally, we found the main advantage of backward correction to be the abilitiy of the method to be integrated into any standard pipeline and outputting good results. Co-teaching requires much more smarts and tuning, but eventually it has better potential to produce higher quality. The most important result of the work is that, of course, it is more than worth it to adapt the models to the instabilities in the training ground truth.

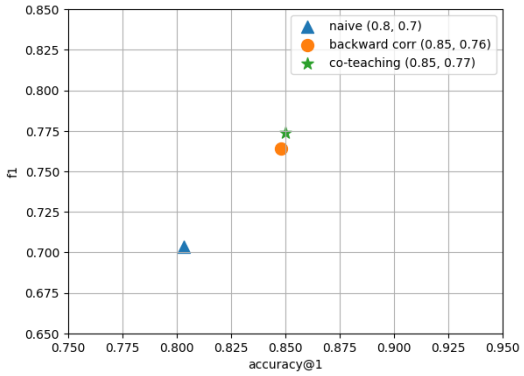

Figure 1: Accuracy vs f1 score for each method averaged by dataset

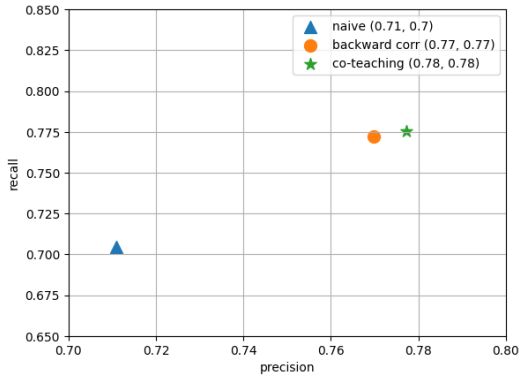

Figure 2: Precision vs recall for each method averaged by dataset

5.1.1 Future Work

An important part of future work is analysing the tuning parameters to get the best metrics as possible for the current methods. As for future methods, one of the key challenges is instance-dependent label noise, which can very well be found in natural environments such as manually labeled datasets. Moreover, it is quite hard to get naturally noise datasets - here in this work we manually artificially noise the existing ones. So, three things: fine tune the parameters to maximum, use naturally noisy datasets and tackle the instance dependent noise problem.

References

[1] Giorgio Patrini et al. “Making deep neural networks robust to label noise: A loss correction approach”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1944–1952.

[2] Bo Han et al. “Co-teaching: Robust training of deep neural networks with extremely noisy labels”. In: *Advances in neural information processing systems* 31 (2018).

[3] Han Xiao, Kashif Rasul, and Roland Vollgraf. “Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms”. In: *arXiv preprint arXiv:1708.07747* (2017).

[4] Alex Krizhevsky, Geoffrey Hinton, et al. “Learning multiple layers of features from tiny images”. In: (2009).

[5] Jiacheng Cheng et al. “Learning with bounded instance and label-dependent label noise”. In: *International conference on machine learning*. PMLR. 2020, pp. 1789–1799.

[6] Corinna Cortes and Vladimir Vapnik. “Support-vector networks”. In: *Machine learning* 20 (1995), pp. 273–297.

[7] Jerome H Friedman. “Greedy function approximation: a gradient boosting machine”. In: *Annals of statistics* (2001), pp. 1189–1232.

[8] Ashish Vaswani et al. “Attention is all you need”. In: *Advances in neural information processing systems* 30 (2017).

[9] Alexey Dosovitskiy et al. “An image is worth 16x16 words: Transformers for image recognition at scale”. In: *arXiv preprint arXiv:2010.11929* (2020).

[10] Jia Deng et al. “Imagenet: A large-scale hierarchical image database”. In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[11] Tongliang Liu and Dacheng Tao. “Classification with noisy labels by importance reweighting”. In: *IEEE Transactions on pattern analysis and machine intelligence* 38.3 (2015), pp. 447–461.

[12] Lorenzo Bruzzone and Mattia Marconcini. “Domain adaptation problems: A DASVM classification technique and a circular validation strategy”. In: *IEEE transactions on pattern analysis and machine intelligence* 32.5 (2009), pp. 770–787.

[13] Arthur Gretton et al. “Covariate shift by kernel mean matching”. In: *Dataset shift in machine learning* 3.4 (2009), p. 5.

[14] Nagarajan Natarajan et al. “Learning with noisy labels”. In: *Advances in neural information processing systems* 26 (2013).

[15] Brendan Van Rooyen et al. “Machine learning via transitions”. In: (2015).

[16] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[17] Devansh Arpit et al. “A closer look at memorization in deep networks”. In: *International conference on machine learning*. PMLR. 2017, pp. 233–242.

[18] Avrim Blum and Tom Mitchell. “Combining labeled and unlabeled data with co-training”. In: *Proceedings of the eleventh annual conference on Computational learning theory*. 1998, pp. 92–100.

[19] Xavier Glorot and Yoshua Bengio. “Understanding the difficulty of training deep feedforward neural networks”. In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings. 2010, pp. 249–256.

[20] Aditya Menon et al. “Learning from corrupted binary labels via class-probability estimation”. In: *International conference on machine learning*. PMLR. 2015, pp. 125–134.

[21] Yann LeCun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.