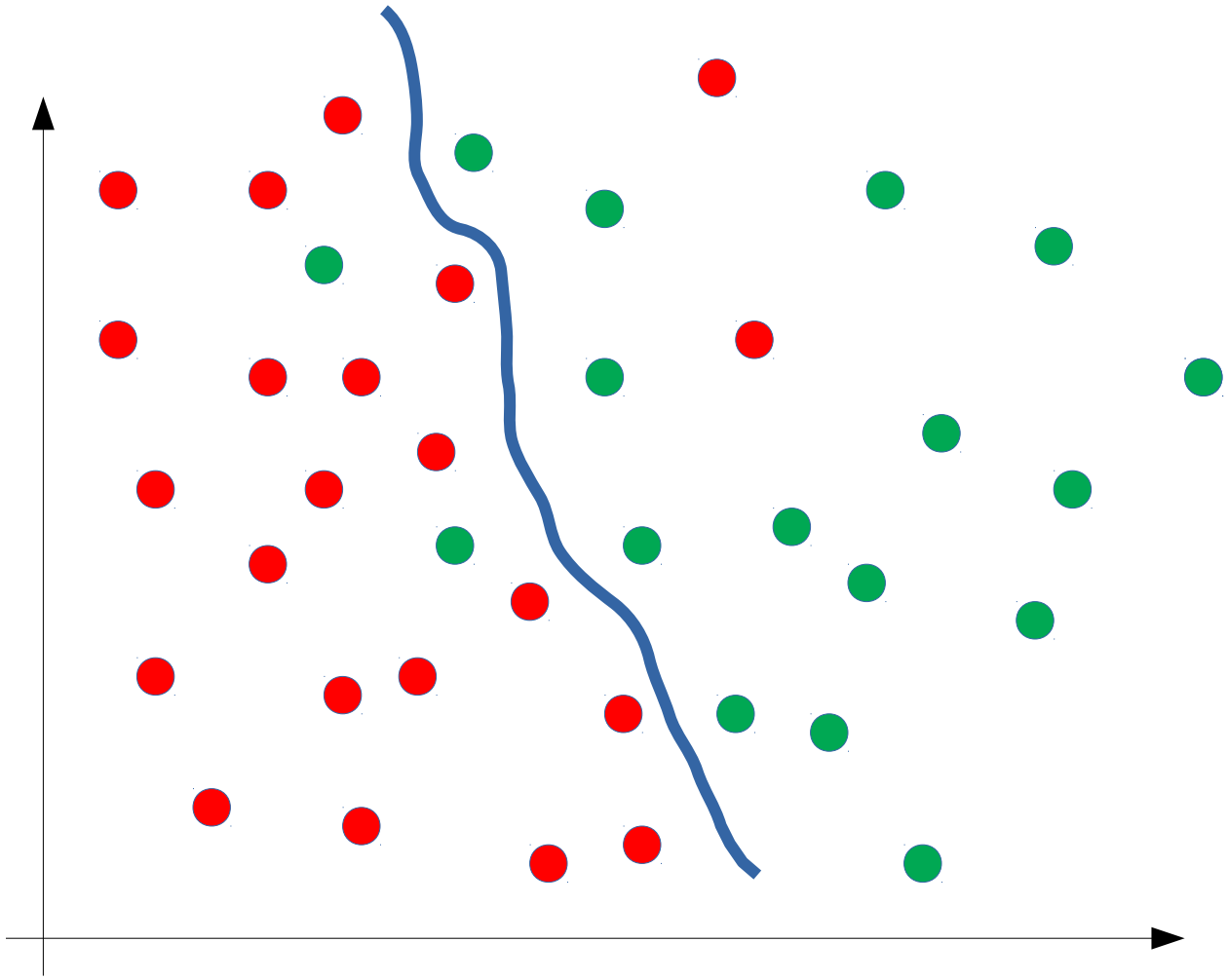
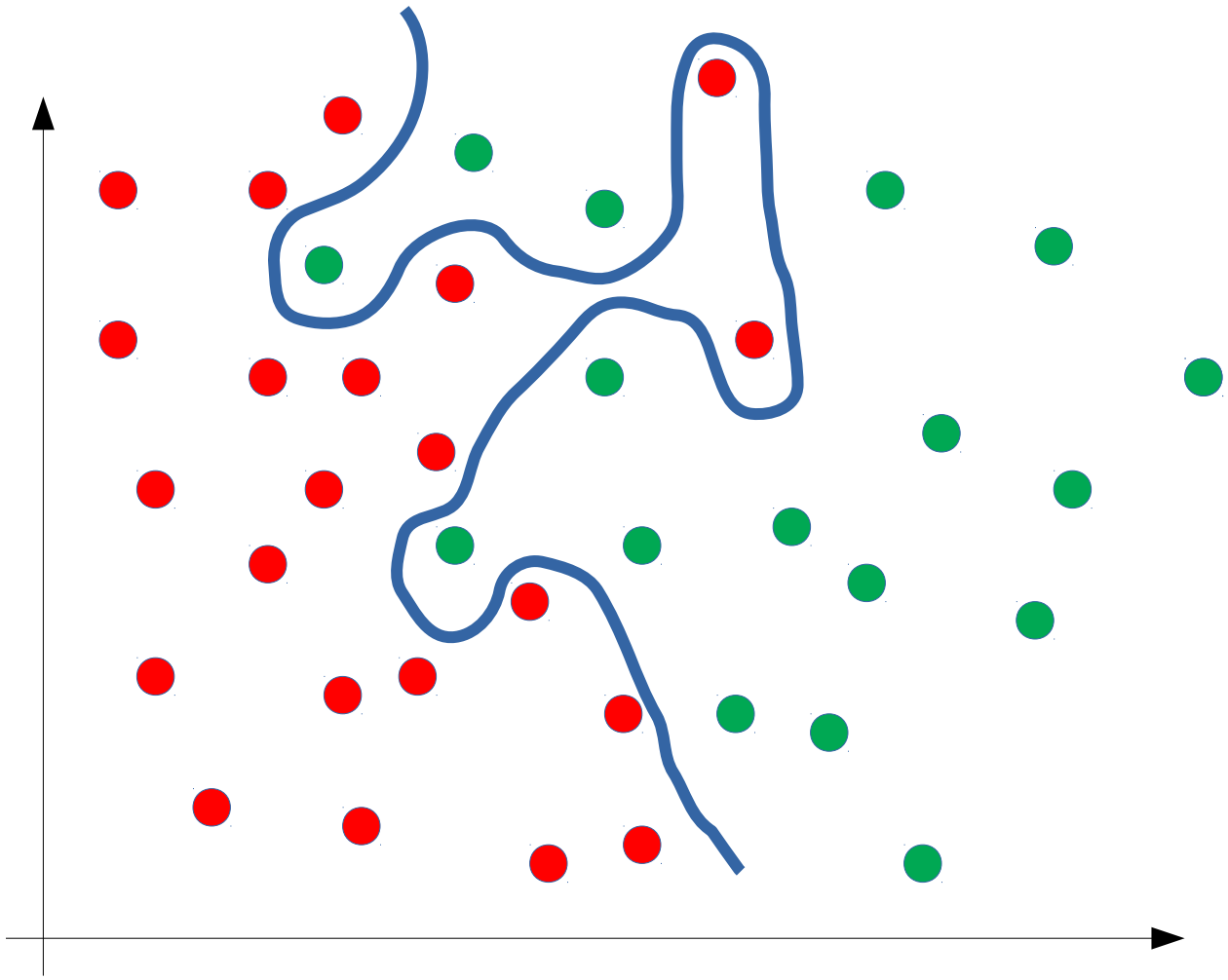
Deep Bilevel Learning

Simon Jenni and Paolo Favaro
University of Bern, Switzerland

Motivation - Good Generalization



Motivation – Generalization



Prior Work

- Provided suitable hyperparameters, CNNs trained with SGD generalizes well even for noisy labels and randomly labeled data [1,2,3].
- Training with weighted examples counters noisy labels [4].
- MaML (Finn et al. [5]) incorporates gradient information of two datasets using second order derivatives.

SGD Update Step (without Bilevel learning)

The update of the parameters at the t-th iteration in SGD (for one mini-batch):

$$\theta^{t+1} = \theta^{t} - \varepsilon \nabla l_i(\theta^{t})$$

Bilevel Learning

Split training set into different mini-batches on each epoch.
We introduce a scalar weight ω_i for each mini-batch in $\mathbf{T}^t$

Bilevel learning then comprises these two tasks:

Upper level: *Find the combination of "final" parameters and weights that minimizes loss on a validation set*

$$\theta'', \acute{\omega} = \mathbf{argmin}_{\theta,\, \omega} \sum_{j \in Vt} l_j(\theta(\omega)) + \mu/2 \cdot |\omega|^2$$

Lower level: *Find the "training set" parameters that minimize loss on training mini-batches*

$$\text{subj. } to\ \theta(\omega) = \mathbf{argmin}_{\theta'} \sum_{i \in Tt} \omega_i l_i(\theta')$$

$$|\omega|_1 = 1$$

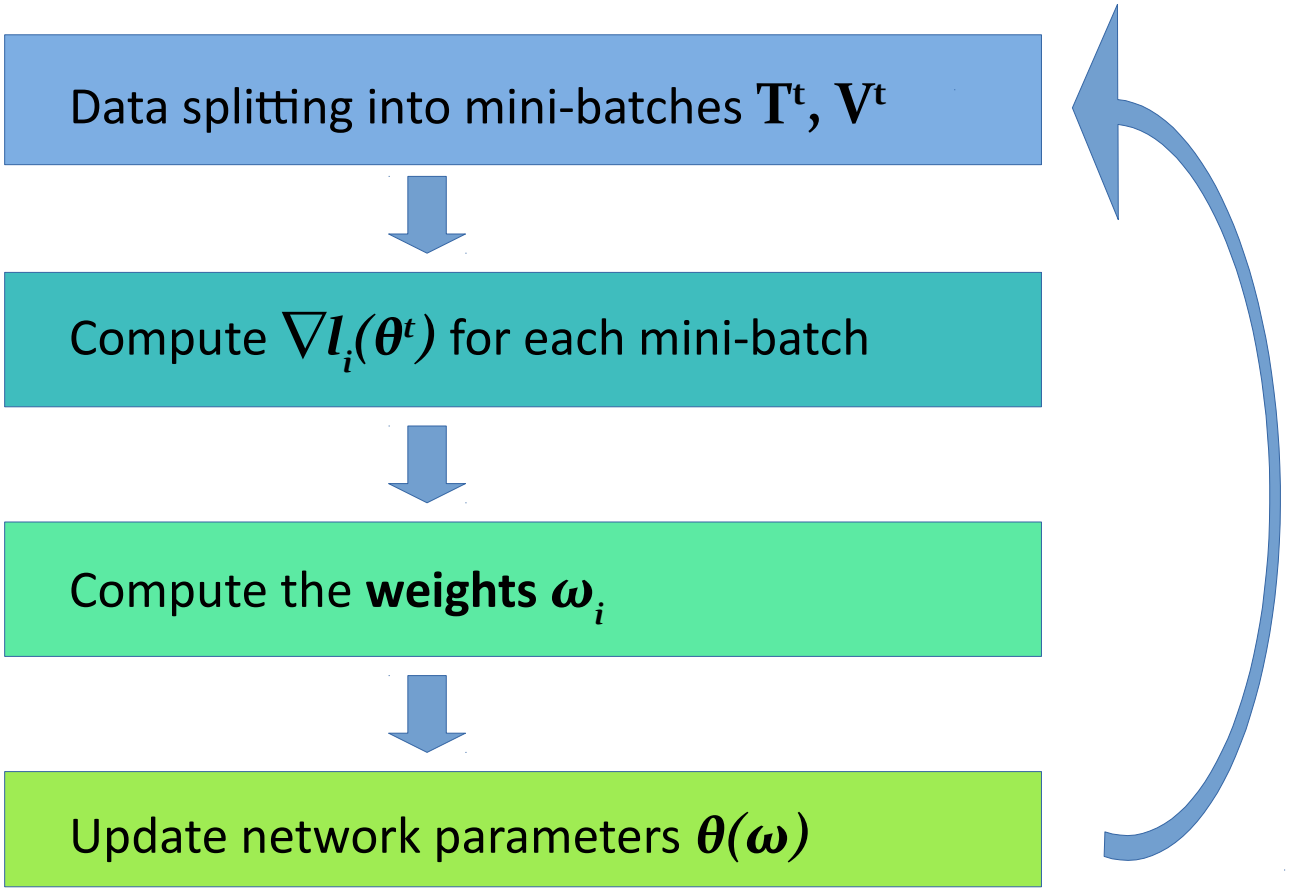
Bilevel Learning

- *Upper-level problem - a performance evaluation on samples from a separate validation set*

- *Lower-level* problem - model parameter optimization on samples from the training set

Implementation

- To implement the method **SGD with momentum** was modified.



Data Splitting

Data splitting into mini-batches T^t, V^t

- m sample pairs $(x^k, y^k)_{k=1,\ldots,m}$
- b disjoint mini-batches B_i
- φ_θ – the model that depends on parameters θ

- The loss function:

$$L_i(\theta) \triangleq \sum_{k \in Bi} L(\varphi_\theta(x^k), y^k))$$

- Split a set of mini-batches into T^t – training set and V^t - validation set

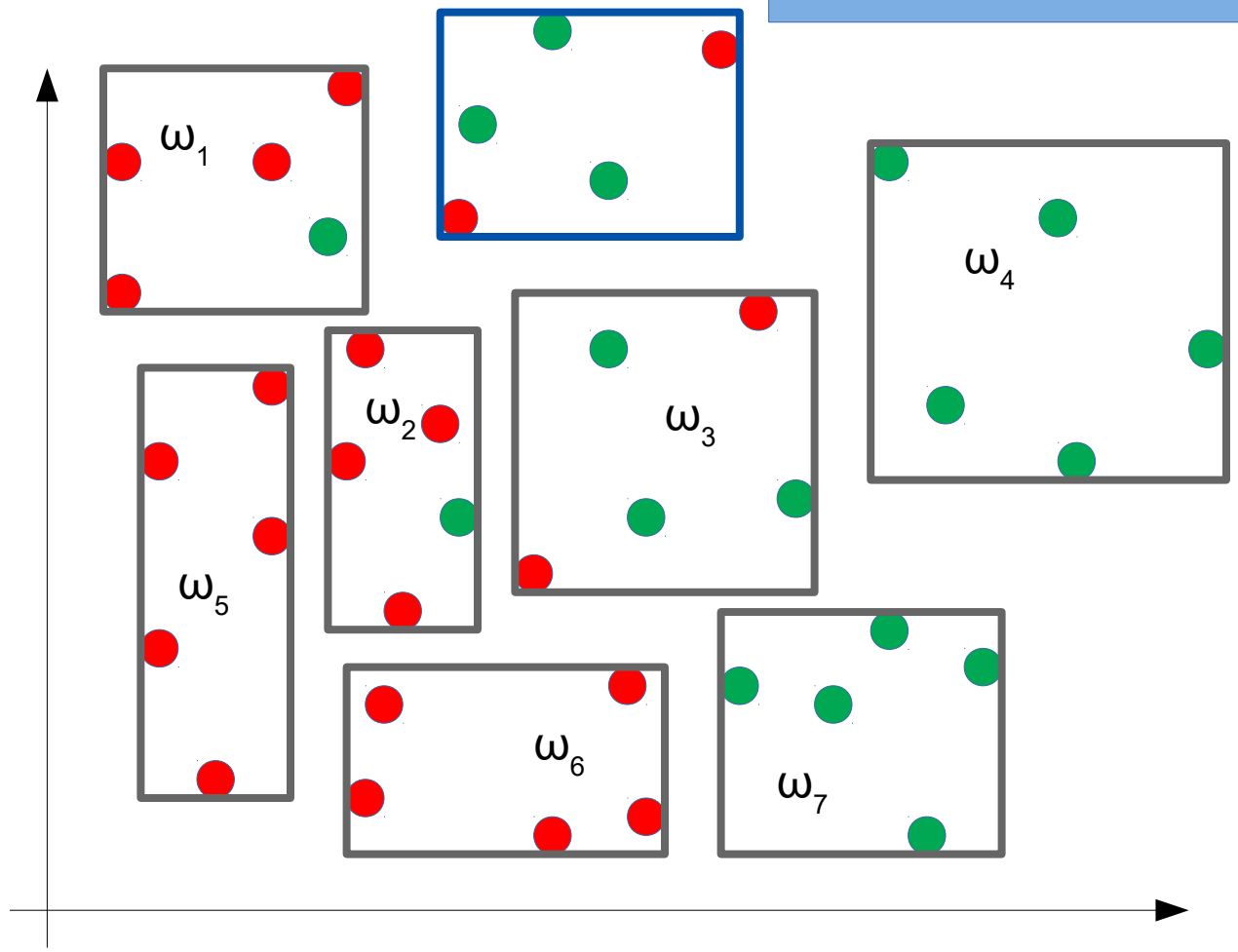
In all experiments V^t is one mini-batch

The **distributions of labels** across the k mini-batches are **identical**.

Bilevel Learning

Validation set

Data splitting into mini-batches T^t, V^t



At each iteration sample the data and conduct gradient descent step.
k-1 mini-batches with weights: $\omega_1, \omega_2, \ldots, \omega_{k-1}$, 1 mini-batch for validation

A Proximal Formulation

First-order Taylor expansion:

$$l_i(\theta) \approx l_i(\theta^t) + \nabla l_i(\theta^t)^T(\theta - \theta^t)$$

A Proximal Formulation

First-order Taylor expansion:

$$l_i(\theta) \approx l_i(\theta^t) + \nabla l_i(\theta^t)^T(\theta - \theta^t)$$

$$\theta^{t+1}, \acute{\omega} = \mathbf{argmin}_{\theta,\,\omega} \sum_{j \in Vt} l_j(\theta^t)^T(\theta(\omega) - \theta^t) + |\theta(\omega) - \theta^t|^2/2\lambda + \mu/2 \cdot |\omega|$$

subj. to $$\theta(\omega) = \mathbf{argmin}_{\theta'} \sum_{i \in Tt} \omega_i [\, l_i(\theta^t) + \nabla l_i(\theta^t)^T(\theta' - \theta^t)] + |\theta' - \theta^t|^2/2\varepsilon$$

$$|\omega|_1 = 1$$

A Proximal Formulation

First-order Taylor expansion:

$$l_i(\theta) \approx l_i(\theta^t) + \nabla l_i(\theta^t)^T(\theta - \theta^t)$$

$$\theta^{t+1}, \acute{\omega} = \mathbf{argmin}_{\theta,\, \omega} \sum_{j \in Vt} l_j(\theta^t)^T(\theta(\omega) - \theta^t) + |\theta(\omega) - \theta^t|^2/2\lambda + \mu/2 \cdot |\omega|$$

subj. to $$\theta(\omega) = \mathbf{argmin}_{\theta'} \sum_{i \in Tt} \omega_i [\, l_i(\theta^t) + \nabla l_i(\theta^t)^T(\theta' - \theta^t)] + |\theta' - \theta^t|^2/2\varepsilon$$

$$|\omega|_1 = 1$$

Update the parameters via:

$$\theta(\omega) = \theta^t - \varepsilon \sum_{i \in Tt} \acute{\omega}_i \nabla l_i(\theta^t)$$

Weights Computation

Compute the **weights** ω_i

$$\acute{\omega} = \mathbf{argmin}_{\theta,\,\omega} \sum_{j \in Vt,\, i \in Tt} - \omega_i \nabla l_j(\theta^t)^T \nabla l_i(\theta^t) + |\sum_{i \in Tt} \omega_i \nabla l_i(\theta^t)|^2/(2\lambda/\varepsilon) + (\mu/2\varepsilon)\cdot|\omega|^2,$$

$$s.t.\ |\omega|_1 = 1$$

Weights Computation

Compute the **weights** ω_i

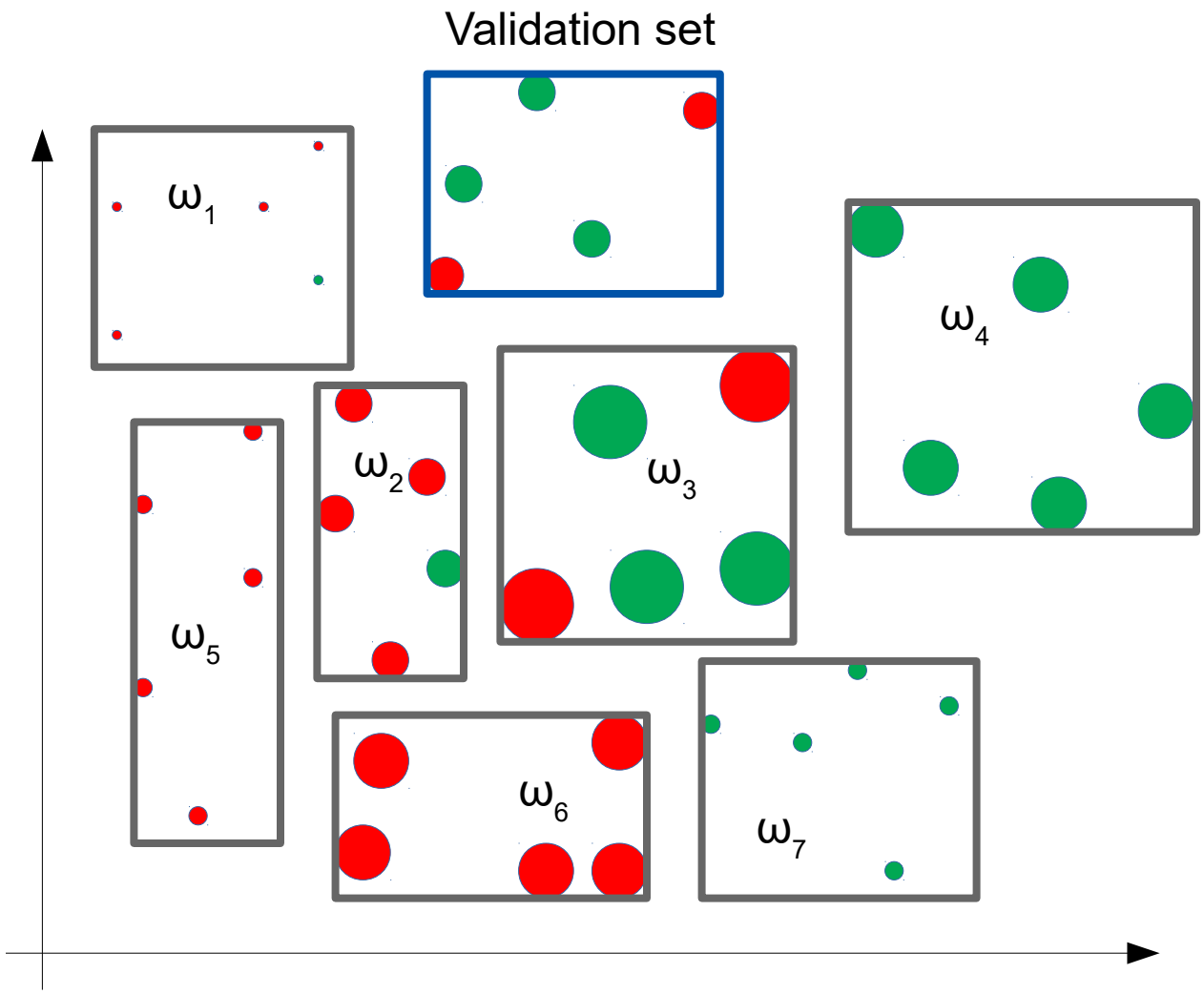
$$\acute{\omega} = \mathbf{argmin}_{\theta,\,\omega} \sum_{j\in Vt,\, i\in Tt} - \omega_i \nabla l_j(\theta^t)^T \nabla l_i(\theta^t) + |\sum_{i\in Tt} \omega_i \nabla l_i(\theta^t)|^2/(2\lambda/\varepsilon) + (\mu/2\varepsilon)\cdot|\omega|^2,$$
$$s.t.\ |\omega|_1 = 1$$

$\lambda' = \lambda/\varepsilon,\ \ \mu' = \mu/\varepsilon,$

Solve the equation for any $i \in T^t$

$$\omega_i \leftarrow \sum_{j\in Vt} \nabla l_j(\theta^t)^T \nabla l_i(\theta^t)/(|\nabla l_j(\theta^t)|^2/(\lambda'+\mu'))$$

Data Splitting – weights optimization



At each iteration sample the data and conduct gradient descent step.
k-1 mini-batches with weights: ω_1, ω_2, … , $\omega_{k-1,}$ 1 mini-batch for validation

Update Step

Update network parameters $\theta(\omega)$

$$\acute{\omega} = \mathbf{argmin}_{\theta,\,\omega} \sum_{j\in Vt,\, i\in Tt} - \omega_i \nabla l_j(\theta^t)^T \nabla l_i(\theta^t) + |\sum_{i\in Tt} \omega_i \nabla l_i(\theta^t)|^2/(2\lambda/\varepsilon) + (\mu/2\varepsilon)\cdot|\omega|^2,$$
$$s.t.\ |\omega|_1 = 1$$

$\lambda' = \lambda/\varepsilon,\ \ \mu' = \mu/\varepsilon,$

Solve the equation for any $i \in T^t$

$$\omega_i \leftarrow \sum_{j\in Vt} \nabla l_j(\theta^t)^T \nabla l_i(\theta^t)/(|\nabla l_j(\theta^t)|^2/(\lambda'+\mu'))$$

$$\acute{\omega} = \omega/|\omega|_1$$

$$\theta(w) = \theta^t - \varepsilon \sum_{i\in Tt} \acute{\omega}_i \nabla l_i(\theta^t)$$

Weights based on similarity to val. set

data set

training mini-batches

$$\sum_{i \in \mathcal{T}^t} \omega_i \ell_i(\theta)$$

validation mini-batches

$$\sum_{j \in \mathcal{V}^t} \ell_j(\theta)$$

$\nabla \ell_i(\theta^t)$

$\nabla \ell_j(\theta^t)$

if gradients agree
the mini-batch weights
are positive and large

mini-batch weights

$$\omega_i \propto \sum_{j \in \mathcal{V}^t} \frac{\nabla \ell_j(\theta^t)^\top \nabla \ell_i(\theta^t)}{|\nabla \ell_i(\theta^t)|^2 + \hat{\mu}}$$

$$\theta = \theta^t - \epsilon \sum_{i \in \mathcal{T}^t} \omega_i \nabla \ell_i(\theta^t)$$

stochastic gradient descent
with mini-batch adaptive weights

Experiments

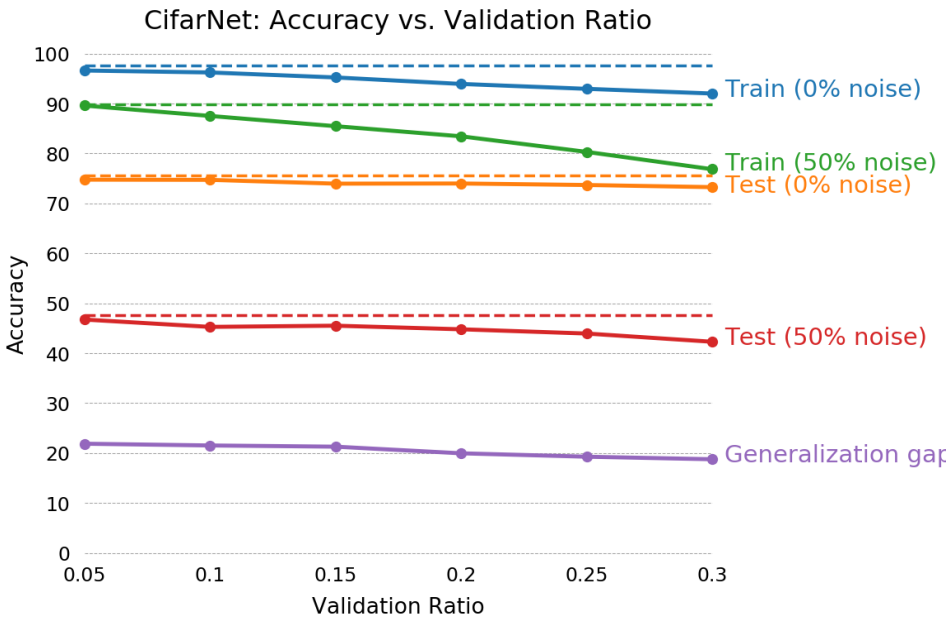
The datasets used for the experiments:

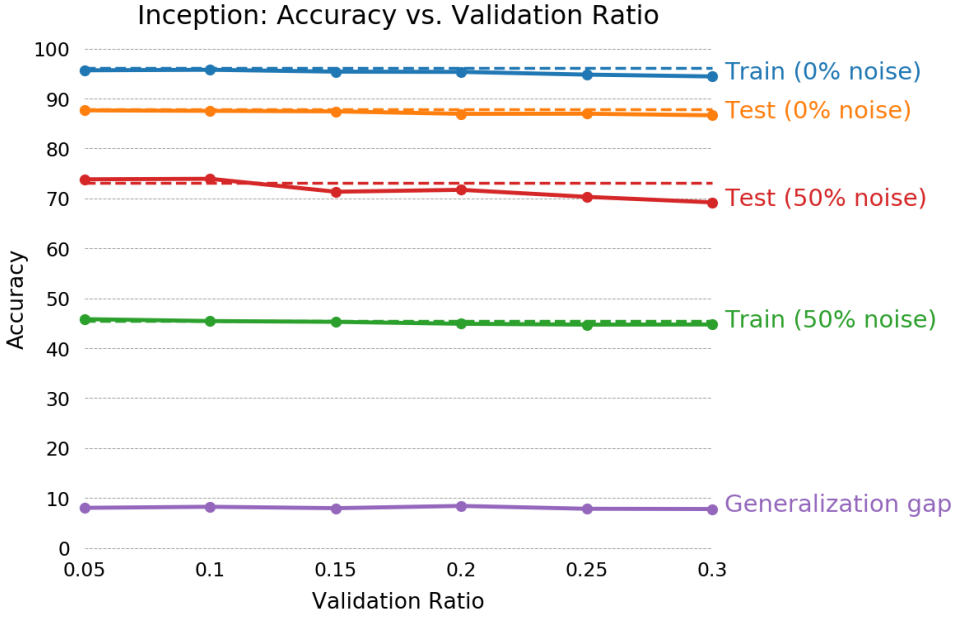
- **CIFAR-10** - 50K training and 10K test images, size 32x32 pixels, 10 classes
- **CIFAR-100** - 50K training and 10K test images, size 32x32 pixels, 100 classes
- **Pascal VOC 2007** - 5011 training and 4952 test images, 20 classes
- **ImageNet** – 1.28M training and 50K test images, 1K classes

AlexNet: Pascal VOC and **ImageNet**.
CifarNet and a **small Inception: CIFAR-10** and **CIFAR-100**.

Ablation Experiments

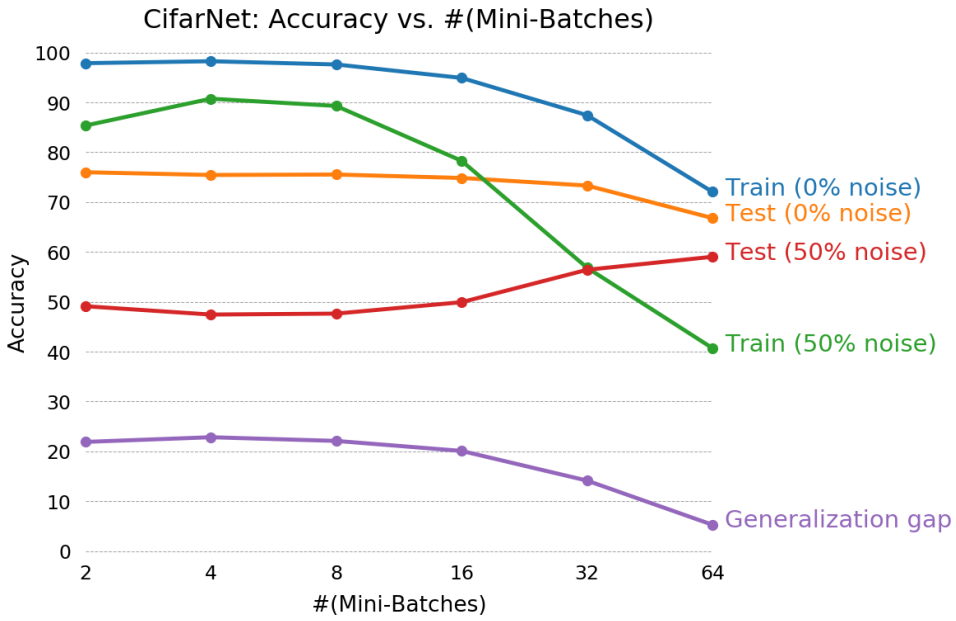
- Dataset: **CIFAR-10**.

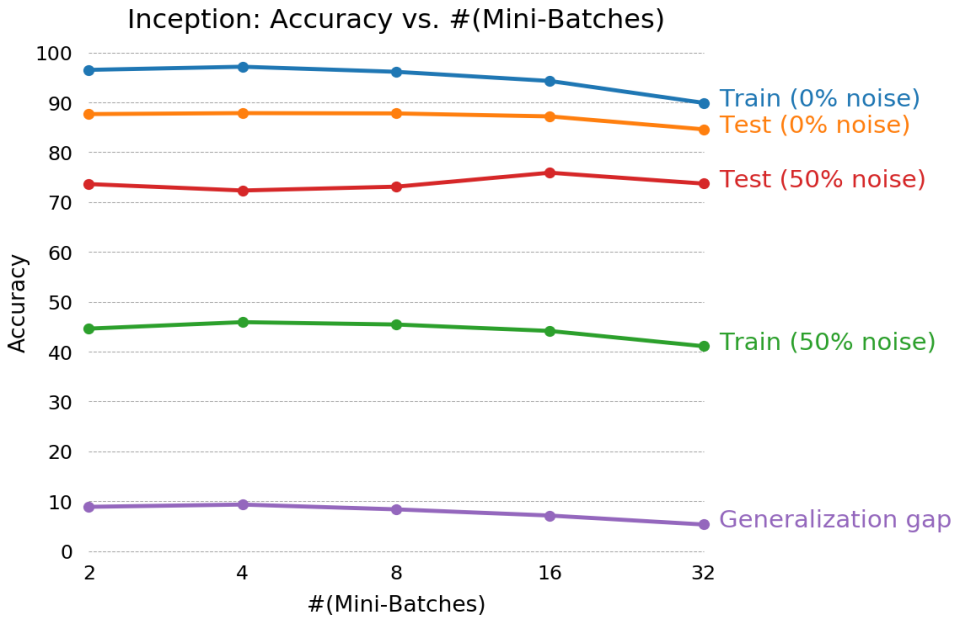




Reduction of a training set size could cause the decreasing of the accuracy.

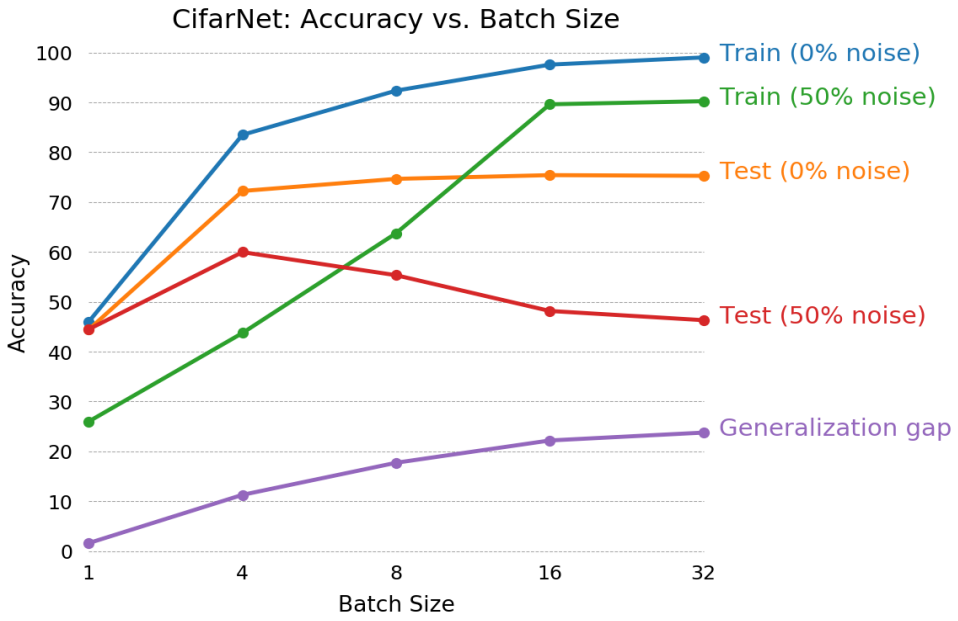
Ablation Experiments

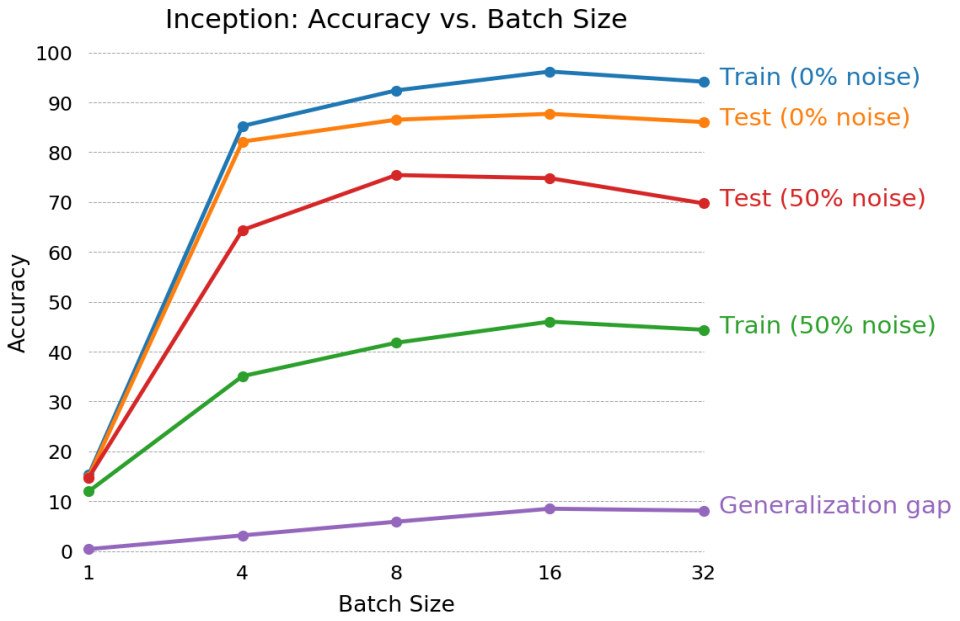




The number of training steps is a constant.
More mini-batches corresponds to smaller batch sizes.

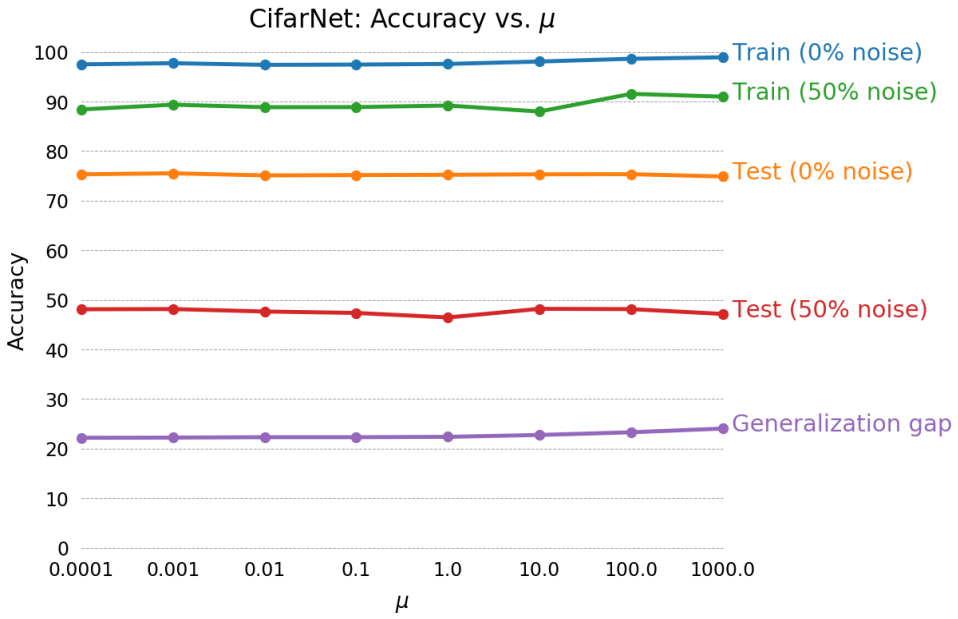
Ablation Experiments

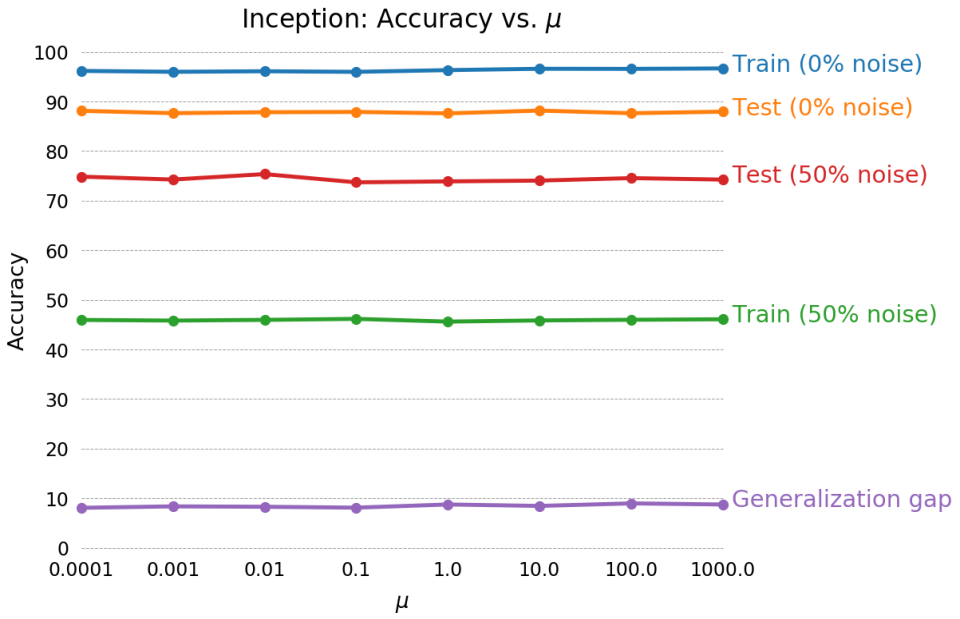




The number of mini-batches is fixed at 8.
Small mini-batch sizes lead to better generalization.

Ablation Experiments





The parameter μ does not seem to have a significant influence on the performance.

Ablations Summary (Clean Data)

Experiment	CifarNet			Inception		
	Train	Test	Gap	Train	Test	Gap
SGD	99.99	75.68	24.31	99.91	88.13	11.78
Baseline Bilevel	97.60	75.52	22.08	96.13	87.78	8.35
No L^1 ($\|\omega\|_1 = 1$)	96.44	74.32	22.12	79.46	77.07	2.39
Not forcing equal label distributions	72.69	68.19	4.50	79.78	78.25	1.53
Allowing different dropout	95.92	74.76	21.16	95.58	87.86	7.72

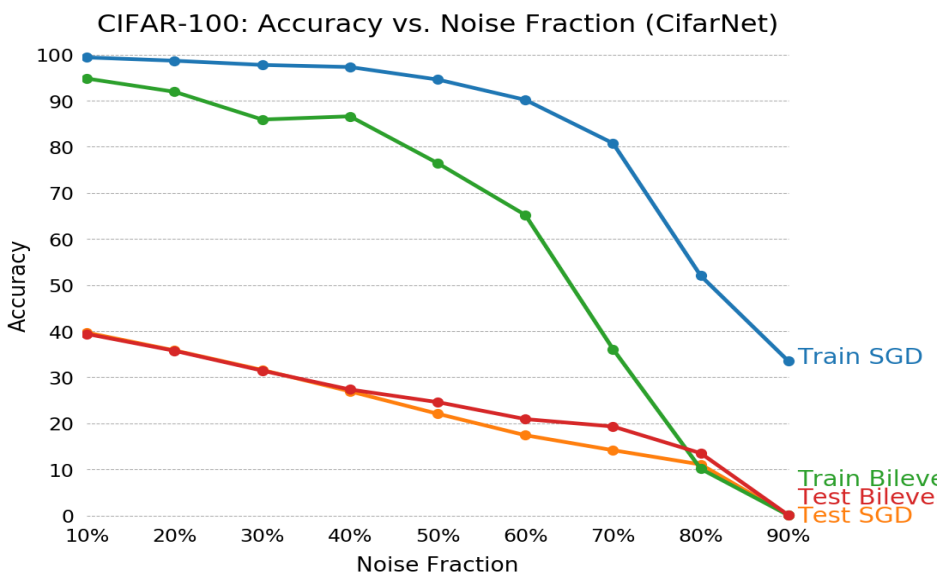
Random Pixel Permutations

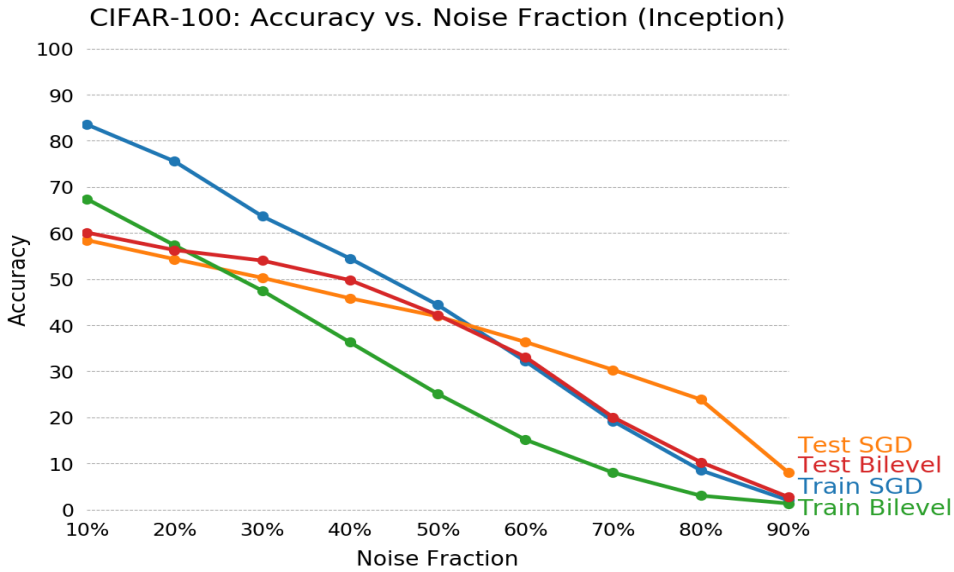
- The setup from [1] – is followed
  (same permutation is applied to all the images in both training and test set)
- Inception network

Model	Train	Test	Gap
SGD	50.0	33.2	16.8
Bilevel	34.8	33.6	1.2

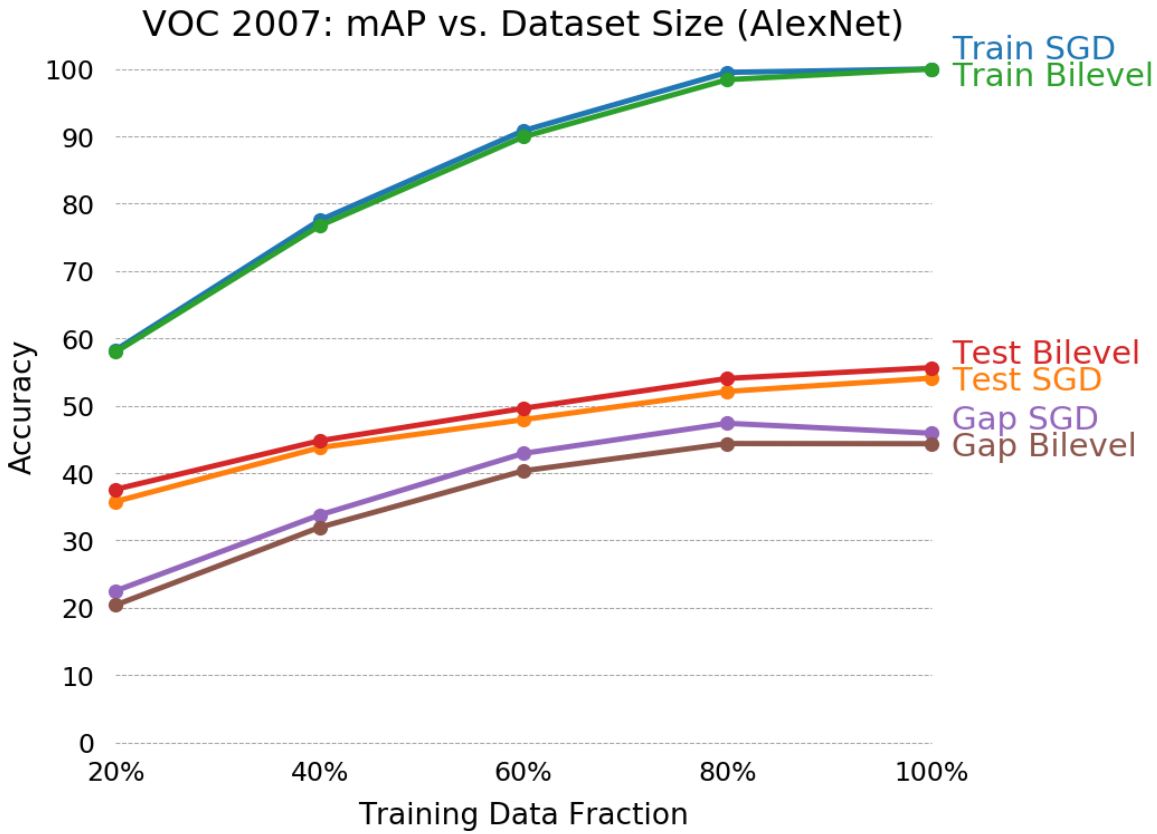
Memorization of Partially Corrupted Labels

The networks are trained with 8 mini-batches, 100 epochs on batches of size 64.





Generalization on Small Datasets



- Training images are randomly cropped to an area between 30% to 100% of the original and then resized to 227x227. mAP is obtained from the average prediction over 10 random crops
- decay the learning rate from 0.01 to 0 and train for 1K epochs
- mini-batches of size 64, 4 mini-batches

Summary

- Bilevel Learning is based on the principles of cross-validation.
- Data is splitted into training(*lower-level*) and validation(*upper-level*) mini-batches.
- A validation set is used to limit the model overfitting.
- Computationally the method resembles stochastic gradient descent.
- The algorithm improves the generalization of the model, especially for noisy data.

Discussion

- The method is computationally expensive.
- Is the accuracy improvement significant?
- How to sample data in mini-batches effectively?

Sources

[1] Zhang, C., Bengio, S., Hardt, M., Recht, B., Vinyals, O.: Understanding deep learning requires rethinking generalization. In: International Conference on Learning Representations (2017)

[2] Zhang, C., Liao, Q., Rakhlin, A., Sridharan, K., Miranda, B., Golowich, N., Poggio, T.: Theory of deep learning iii: Generalization properties of sgd. Tech. rep., Center for Brains, Minds and Machines (CBMM) (2017)

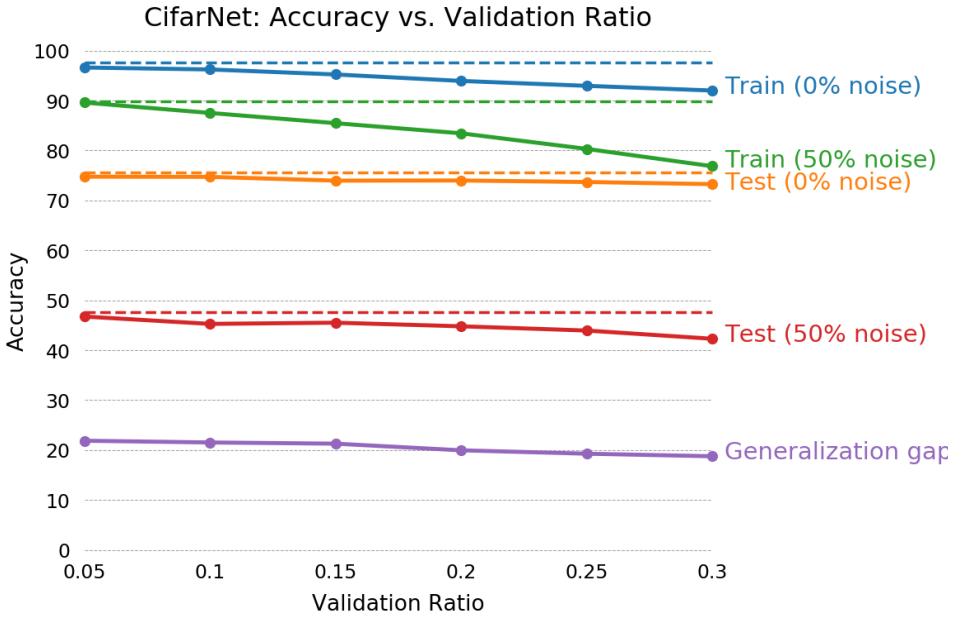
[3] Rolnick, D., Veit, A., Belongie, S., Shavit, N.: Deep learning is robust to massive label noise. arXiv preprint arXiv:1705.10694 (2017)

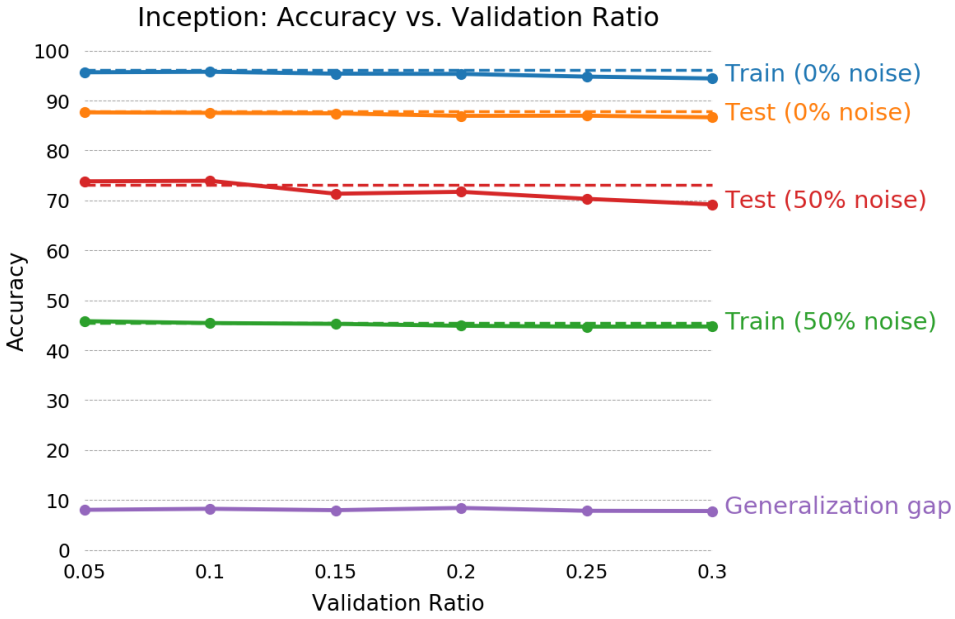
[4] Jiang, L., Zhou, Z., Leung, T., Li, L.J., Fei-Fei, L.: Mentornet: Regularizing very deep neural networks on corrupted labels. arXiv preprint arXiv:1712.05055 (2017)

[5] Finn, C., Abbeel, P., Levine, S.: Model-agnostic meta-learning for fast adaptation of deep networks. arXiv preprint arXiv:1703.03400 (2017)

Ablation Experiments

- Dataset: **CIFAR-10**
- **8** mini-batches of size **128**
- μ **= 0.01** and **λ**= **1**
- **Single dropout layer**, same dropping in all mini-batches
- **200** epochs
- **Standard augmentations** only for the **Inception** network
- SGD with momentum of **0.9** and an initial learning rate of **0.01** in the case of CifarNet and **0.1** for Inception.



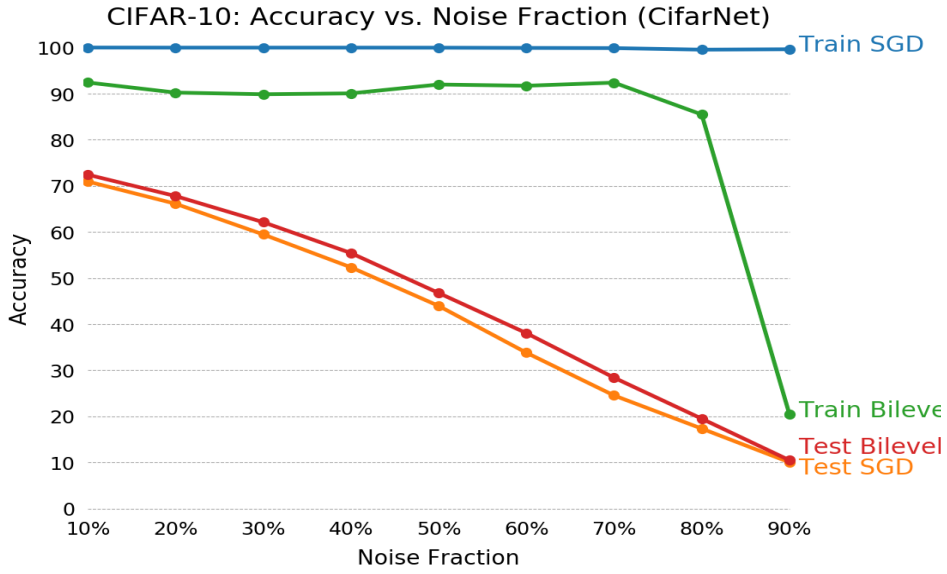


Reduction of a training set size could cause the decreasing of the accuracy.

Ablations Summary

Experiment	CifarNet					Inception				
	Clean			50% Random		Clean			50% Random	
	Train	Test	Gap	Train	Test	Train	Test	Gap	Train	Test
SGD	99.99	75.68	24.31	96.75	45.15	99.91	88.13	11.78	65.06	47.64
Baseline	97.60	75.52	22.08	89.28	47.62	96.13	87.78	8.35	45.43	73.08
L^1	96.44	74.32	22.12	95.50	45.79	79.46	77.07	2.39	33.86	62.16
ω per Layer	97.43	74.36	23.07	81.60	49.62	90.38	85.25	5.13	81.60	49.62
Sampling	72.69	68.19	4.50	16.13	23.93	79.78	78.25	1.53	17.71	27.20
Dropout	95.92	74.76	21.16	82.22	49.23	95.58	87.86	7.72	44.61	75.71

Memorization of Partially Corrupted Labels



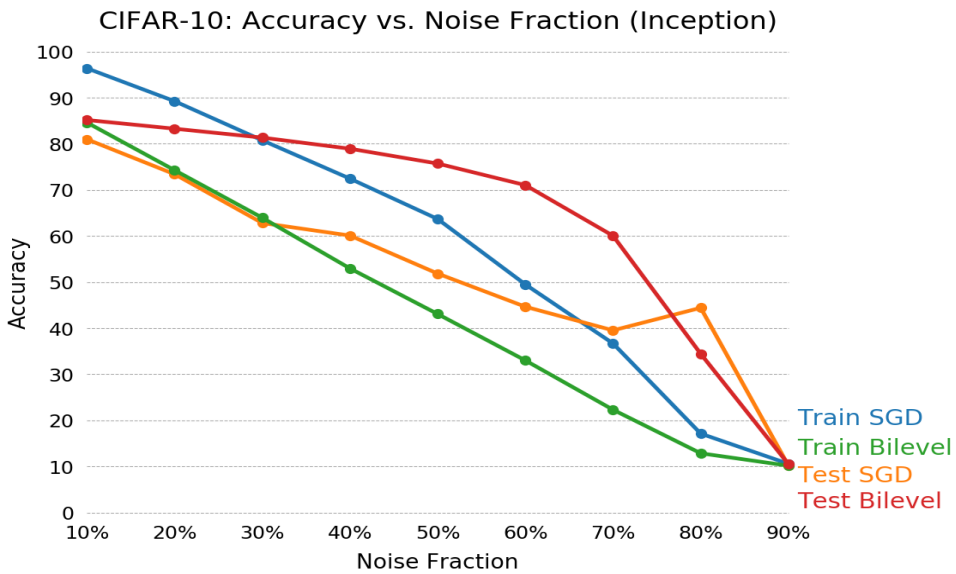


Table 3. Comparison to state-of-the-art regularization techniques and methods for dealing with label noise on 40% corrupted labels.

Method	Ref.	Network	CIFAR-10	CIFAR-100
Reed *et al.* [27]	[14]	ResNet	62.3%	46.5%
Golderberger *et al.* [11]	[14]	ResNet	69.9%	45.8%
Azadi *et al.* [2]	[2]	AlexNet	75.0%	-
Jilang *et al.* [14]	[14]	ResNet	76.6%	56.9%
Zhang *et al.* [38]	-	PreAct ResNet-18	88.3%	56.4%
Standard SGD	-	PreAct ResNet-18	69.6%	44.9%
Dropout ($p = 0.3$) [30]	-	PreAct ResNet-18	84.5%	50.1%
Label Smoothing (0.1) [32]	-	PreAct ResNet-18	69.3%	46.1%
Bilevel	-	PreAct ResNet-18	87.0%	59.8%
Bilevel + [38]	-	PreAct ResNet-18	89.0%	61.6%

Modeling Realistic Label Noise on ImageNet

- Predicted labels of a pre-trained AlexNet were used to model realistic label noise (for ImageNet).
- To obtain a high noise level - leave dropout active when making the predictions on the training set ~ 44% label noise.
- Retrain an AlexNet from scratch on those labels using standard SGD and our bilevel optimizer.

Method	44% Noise	Clean
SGD	50.75%	57.4%
Bilevel	52.69%	58.2%