Learning is a Bilevel Optimization problem

Workshop ILLS/DATAIA

Marco Pedersoli

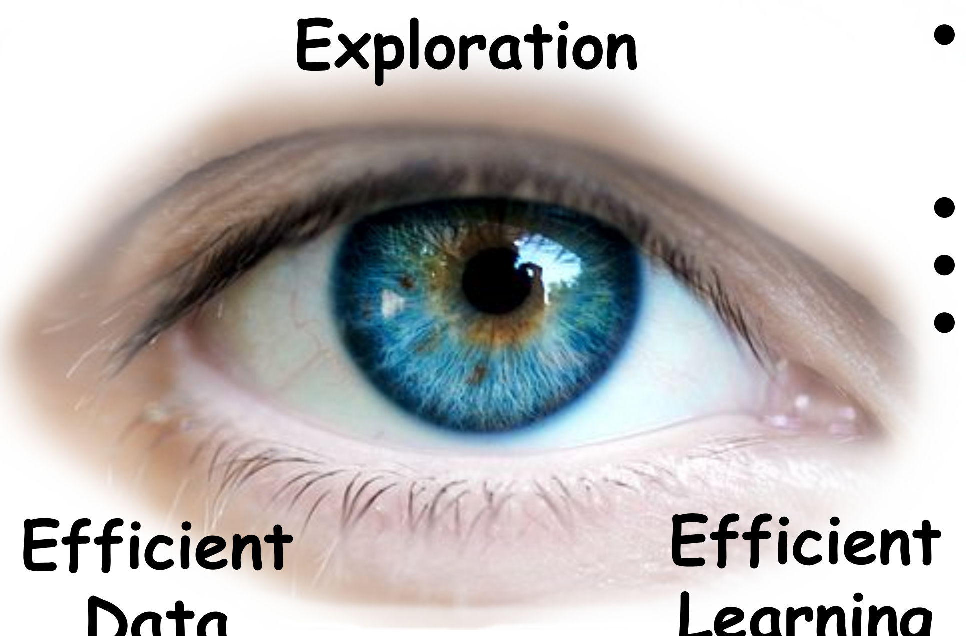
26/05/2023

LIVIA
LABORATOIRE D'IMAGERIE, DE VISION ET D'INTELLIGENCE ARTFICIELLE

ÉTS Le génie pour l'industrie
ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
Université du Québec

ILLS
International Laboratory on Learning Systems

My Research



- *Associate prof at ÉTS Montréal*
- *Industrial Research Co-Chair in Building Automation with Deep Learning*
- *Member of LIVIA & ILLS*
- *Affective Computing Lab*
- *Collaboration with several companies:*
  - *Distech Controls, Ericsson, ServiceNow, Ubisoft, Teledyne Dalsa, MDA, CAE*

Learning

$$w^* = \arg\min_{w} \mathcal{L}(f(\mathcal{X}, w), \mathcal{Y})$$

$\mathcal{X} =$ Samples

$\mathcal{Y} =$ Annotations

$w =$ Model parameters

$\mathcal{L} =$ Loss function

Model Selection

$$\theta^* = \arg\min_w \mathcal{L}(f_\theta(\mathcal{X}_{val}, w^*), \mathcal{Y}_{val})$$

$$w^* = \arg\min_w \mathcal{L}(f_\theta(\mathcal{X}, w), \mathcal{Y})$$

$\mathcal{X} =$ Samples

$\theta =$ Model params

$\mathcal{Y} =$ Annotations

$w =$ Model parameters

$\mathcal{L} =$ Loss function

Regularization

$$\omega^* = \arg\min_{\omega} \mathcal{L}(f(\mathcal{X}_{val}, w^*), \mathcal{Y}_{val})$$

$$w^* = \arg\min_{w} \mathcal{L}(f(\mathcal{X}, w), \mathcal{Y}) + \mathcal{R}(\omega)$$

$\mathcal{X} =$ Samples

$\mathcal{Y} =$ Annotations

$w =$ Model parameters

$\mathcal{L} =$ Loss function

$\omega =$ Reg. hyper

Latent Variable Estimation

$$\mathcal{Y}^* = \arg\min_{\mathcal{Y}} \mathcal{L}(f(\mathcal{X}_{val}, w^*), \mathcal{Y}_{val})$$
$$w^* = \arg\min_{w} \mathcal{L}(f(\mathcal{X}, w), \mathcal{Y})$$

$\mathcal{X} =$ Samples

$\mathcal{Y} =$ Annotations

$w =$ Model parameters

$\mathcal{L} =$ Loss function

Bilevel Optimization

follower $\Rightarrow$ $\theta^* = \arg\min_{\theta} \mathcal{L}_{val}(w^*, \mathcal{X}_{val}, \mathcal{Y}_{val})$

leader $\Rightarrow$ $w^* = \arg\min_{w} \mathcal{L}(w, \mathcal{X}, \mathcal{Y}, \theta)$

$\mathcal{X} =$ Samples

$\mathcal{Y} =$ Annotations

$w =$ Model parameters

$\mathcal{L} =$ Loss function

$\theta =$ any parameter or latent var.

Bilevel Optimization

Fix θ^*

$$\theta^* = \arg\min_{\theta} \mathcal{L}_{val}(w^*, \mathcal{X}_{val}, \mathcal{Y}_{val})$$

$$w^* = \arg\min_{w} \mathcal{L}(w, \mathcal{X}, \mathcal{Y}, \theta)$$

Fix w^*

- Optimal solution:
  - Double iteration → Very expensive
- Approximations:
  - Trade-off between quality of solution and speed

Bilevel Optimization

- **Model Selection:**
  - Type of convolutions
  - Number of layers, neurons, etc..
  - *Pooling configuration*
- **Regularization:**
  - L1, L2 regularization
  - Early stopping, Batch size
  - *Data Augmentation*
- **Learning with Latent Variables:**
  - Semi-supervised Learning
  - *Temporal localization in videos*
  - *Weakly-supervised Object detection*

Neural Architecture Search (NAS) for CNN pooling

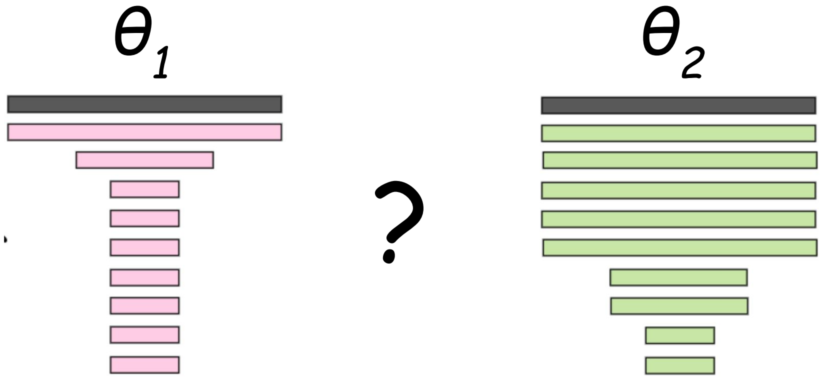
Mehraveh Javan, Matthew Toews, Marco Pedersoli, "Balanced Mixture of Models for Optimal CNN pooling", in AutoML23.

NAS for CNN pooling:
Bilevel Optimization

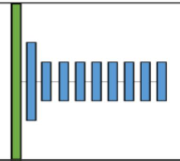
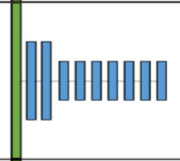
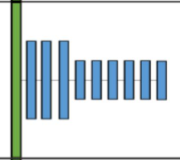
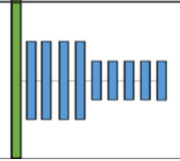
$$\theta^* = \arg\min_{\theta} \mathcal{L}(f_{\color{red}\theta}(\mathcal{X}_{val}, w^*), \mathcal{Y}_{val})$$

$$w^* = \arg\min_{w} \mathcal{L}(f_{\color{red}\theta}(\mathcal{X}, w), \mathcal{Y})$$

θ^* = Pooling configuration



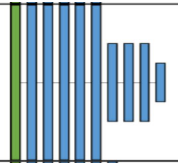
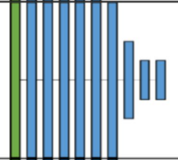
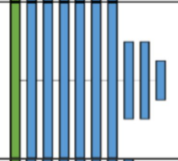
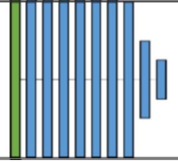
NAS for CNN pooling:
Benchmark on CIFAR10

Architecture	Feature map size	Accuracy
1		87.45
2		87.69
3		87.89
4		88.6

ResNet20

Default: 90.52%
Best: 92.01%

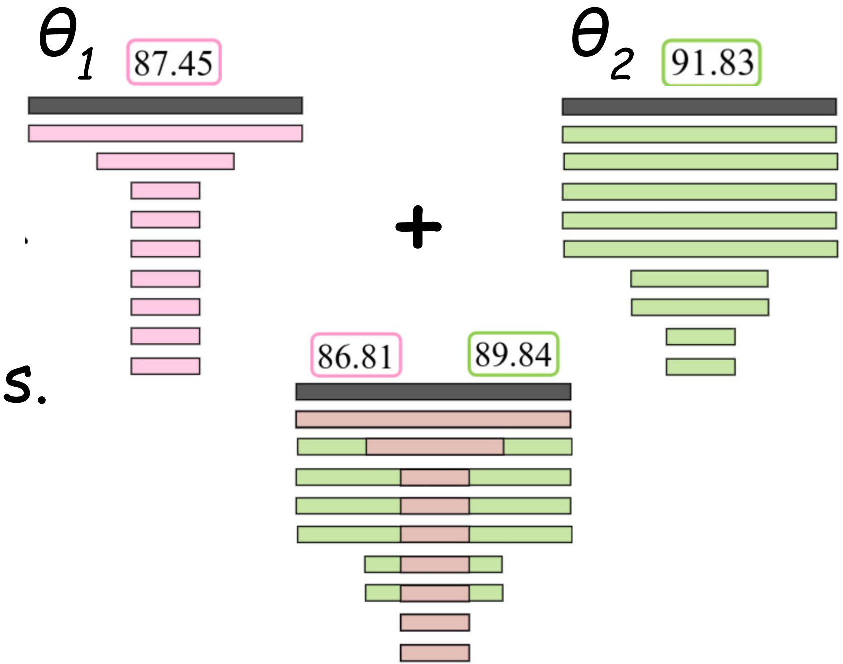
...

Architecture	Feature map size	Accuracy
33		90.96
34		92.01
35		90.47
36		89.99

NAS for CNN pooling:
Challenges

- Non-differentiable
- Weight sharing → Interference between configs.

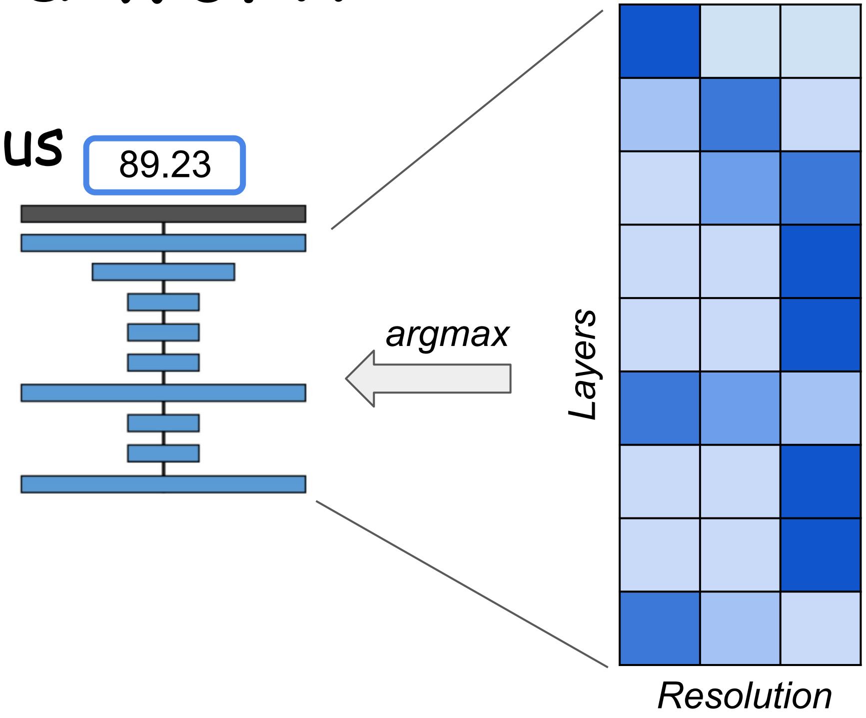
θ = Pooling configuration



NAS for CNN pooling:
Related work

DARTS:
Relaxes pooling into a continuous problem: SuperNet

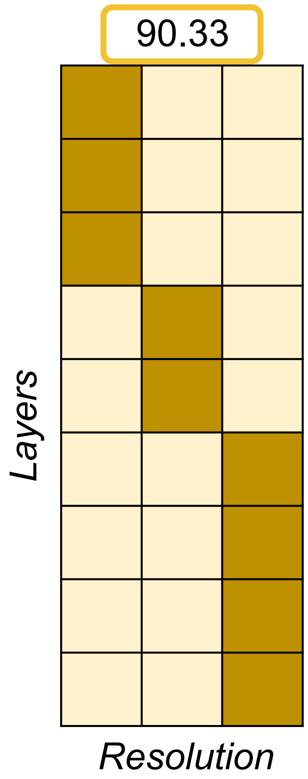
- Memory hungry→ *use always all configurations*
- Multiple paths activated → *interference*
- In practice: *it does not work!!!*



Hanxiao Liu, Karen Simonyan, Yiming Yang, "DARTS: Differentiable Architecture Search", ICLR 2019.

NAS for CNN pooling:
Related work

SPOS (single path one shot):
- Samples uniformly a single path during training
- Architecture selection after training by evaluating SuperNet performance
  - *Less memory*, but still interference between configs
  - Uniform sampling *avoids biases* towards wrong configs
  - Works, but *far from optimal!*



Hanxiao Liu, Karen Simonyan, Yiming Yang, "DARTS: Differentiable Architecture Search", ICLR 2019.
Guo, Z., Zhang, X., Mu, H., Heng, W., Liu, Z., Wei, Y., and Sun, J., "Single path one-shot neural architecture search with uniform sampling", ECCV 2020.

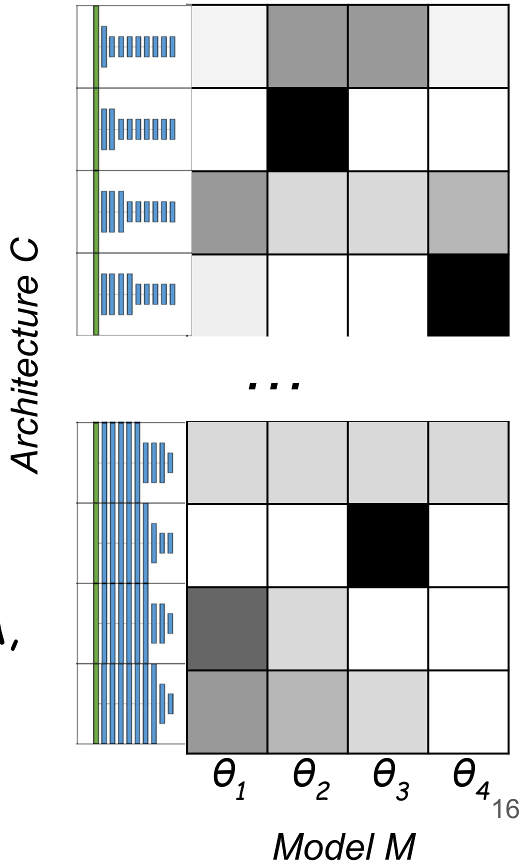
NAS for CNN pooling:

Our Method

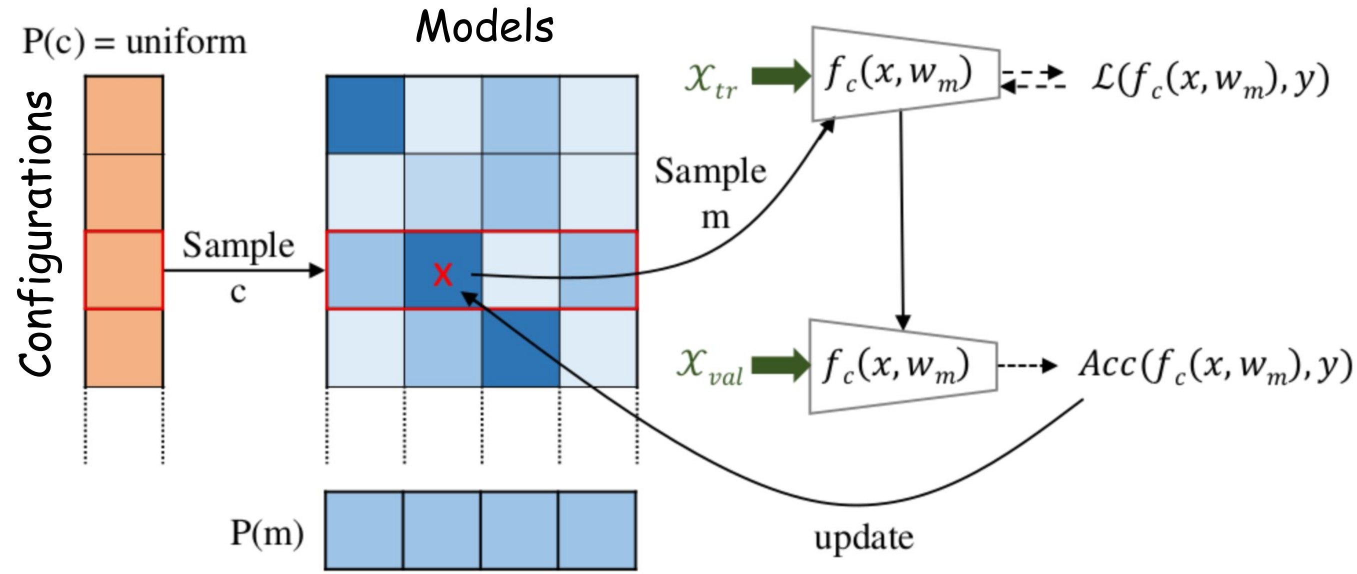
Balanced Mixture of SuperNets

- Sample uniformly on $C \rightarrow$
  No bias during training
  Optimal configuration chosen after

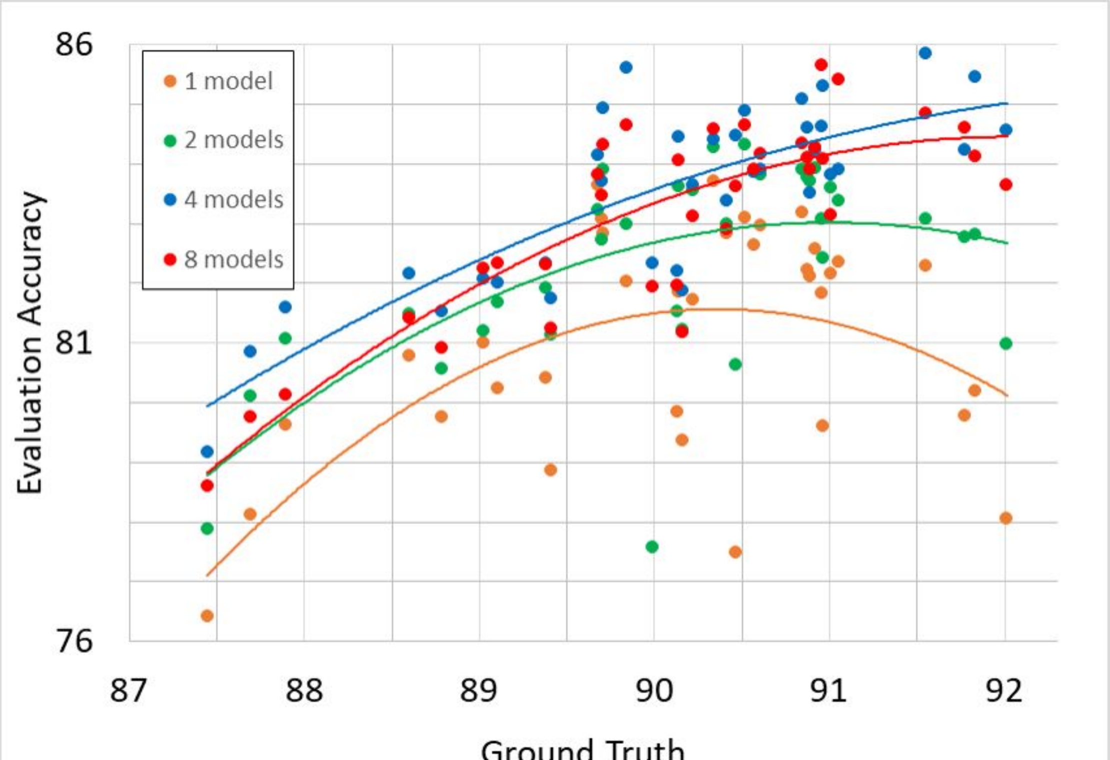
- Multiple Models $\rightarrow$
  - Each architecture config. C is associated to Model M, but same marginal probability for each model
  - Reduces interference between different configs.

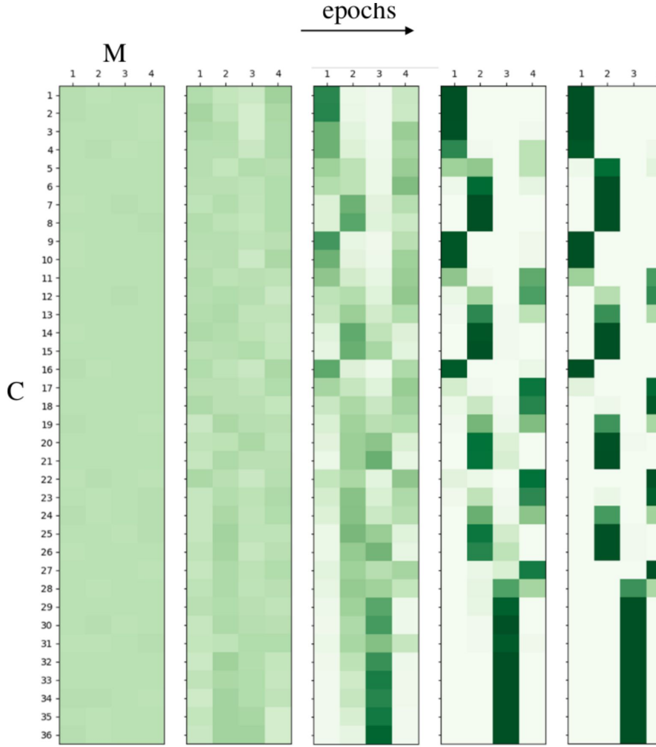


NAS for CNN pooling:
Balances Mixtures of SuperNets



NAS for CNN pooling:
Evaluation CIFAR 10





NAS for CNN pooling:
Evaluation on CIFAR10

NAS Method	Architecture	Accuracy	Training (GPU hours
DARTS + GAEA	Fig. 4a	89.12 ± 0.1	12
DARTS	Fig. 4a	89.23 ± 0.08	12
SBE + Unif. Smp.	[1,6,3]	90.13 ± 0.06	2.5
SPOS	[4,2,4]	90.34 ± 0.12	1.5
MCTS UCB	[4,2,4]	90.34 ± 0.12	2.5
SBE	[2,3,5]	90.42 ± 0.08	2
Default	[4,3,3]	90.52 ± 0.10	-
MCTS UCB + Unif. Smp.	[4,4,2]	90.85 ± 0.09	2.5
Balanced Mixtures (Ours)	[5,3,2]	$\mathbf{91.55 \pm 0.08}$	6
Best conf. (Bruteforce)	[7,1,2]	92.01 ± 0.12	98

Balanced Mixtures is the only model that gets close the the optimal pooling configuration.

NAS for CNN pooling: ImageNet & Food 101

Models	top-1 Arch.	top-1	top-3 Best	Best Arch.	Accuracy
Default	[2,2,2,2]	68.32 ± 0.24	NA	[3,4,6,3]	84.00 ± 0.10
M = 1	[1,3,1,3]	62.21 ± 0.26	65.91 ± 0.21	[6,4,5,1]	84.24 ± 0.09
M = 2	[3,1,1,3]	62.56 ± 0.18	68.32 ± 0.24	[4,5,6,1]	84.34 ± 0.18
M = 4	[5,1,1,1]	65.88 ± 0.24	66.12 ± 0.18	[8,3,3,2]	84.35 ± 0.14
M = 8	[2,3,2,1]	64.81 ± 0.11	66.12 ± 0.23	[9,4,2,1]	84.73 ± 0.09

- On ImageNet best model is the original pooling configuration because ResNet is optimized on it!
- On Food 101 more models and more improvement.

Automatic Data Augmentation

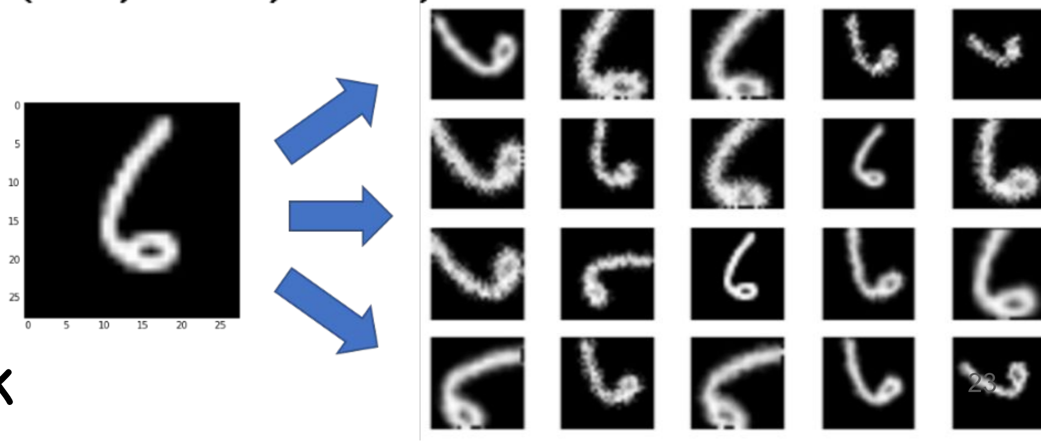
Saypraseuth Mounsaveng, Issam Laradji, Ismail Ben Ayed, David Vazquez, Marco Pedersoli, "Learning data augmentation with online bilevel optimization for image classification", WACV2021

Automatic DA
Bilevel Optimization

$$\theta^* = \arg\min_{\theta} \mathcal{L}(f(\mathcal{X}_{val}, w^*), \mathcal{Y}_{val})$$

$$w^* = \arg\min_{w} \mathcal{L}(f(\mathcal{T}_{\theta}(\mathcal{X}), w), \mathcal{Y})$$

- More challenging than Model Selection because Θ does not appear in the upper-optimisation



T = transformation Network

Automatic DA
Previous Work

$$\theta^* = \arg\min_{\theta} \mathcal{L}(f(\mathcal{X}_{val}, w^*), \mathcal{Y}_{val})$$
$$w^* = \arg\min_{w} \mathcal{L}(f(\mathcal{T}_{\theta}(\mathcal{X}), w), \mathcal{Y})$$

Autoaugment optimizes the bilevel objective by sampling an augmentation policy θ and estimating $\nabla_{\theta}\mathcal{L}(f(\mathcal{X}_{val}, w^*), \mathcal{Y}_{val})$ with RL.

Very slow: a complete training for each inner iteration!

Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, Quoc V. Le, AutoAugment: Learning Augmentation Policies from Data,, CVPR 2019.

Automatic DA
Previous Work

$$\theta^* = \arg\min_{\theta} \mathcal{L}(f(\mathcal{X}_{val}, w^*), \mathcal{Y}_{val})$$
$$w^* = \arg\min_{w} \mathcal{L}(f(\mathcal{T}_{\theta}(\mathcal{X}), w), \mathcal{Y})$$

Randaugment optimises only the magnitude of the transformations θ trying several values
Suboptimal and slow: still a strong baseline

Ekin D. Cubuk, Barret Zoph, Jonathon Shlens, Quoc V. Le, "RandAugment: Practical automated data augmentation with a reduced search space", NeurIPS 2020.

Automatic DA

Our approach

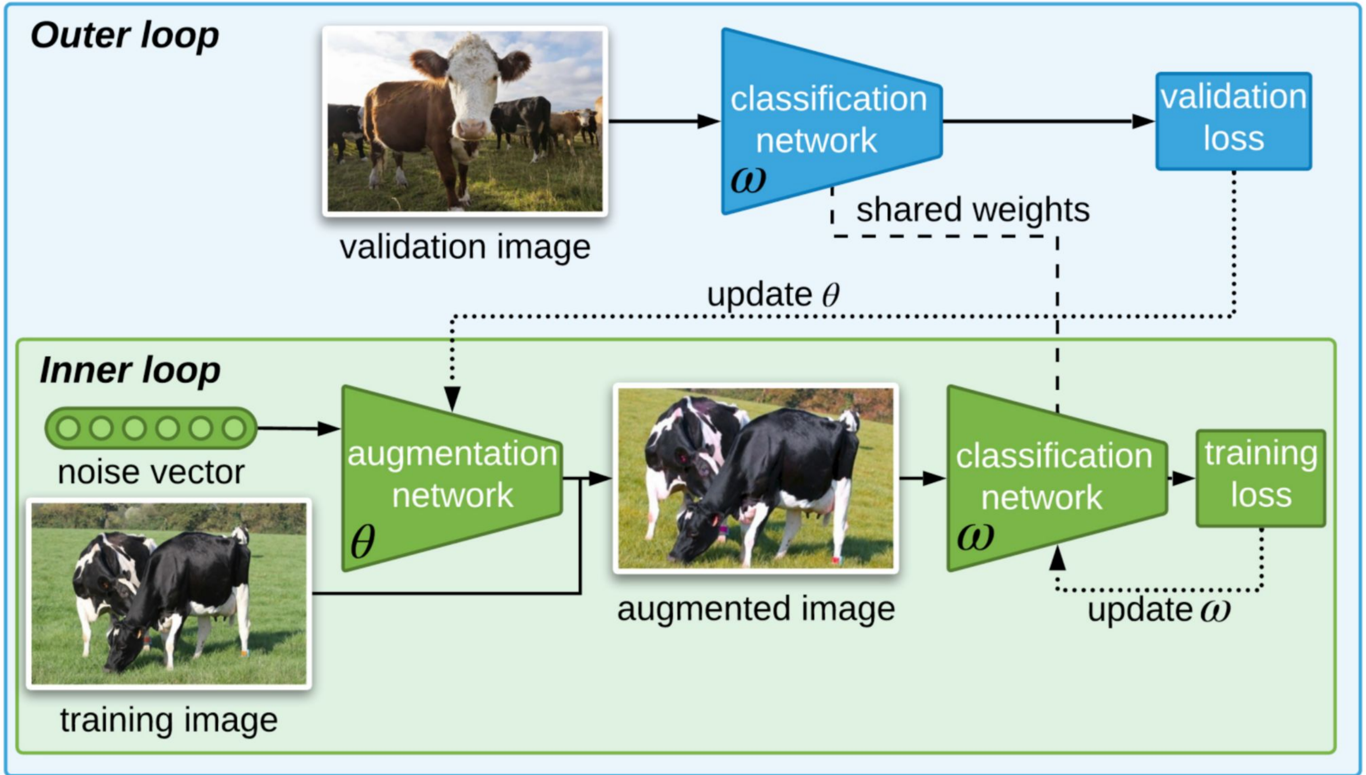
$$\theta^* = \arg\min_{\theta} \mathcal{L}(f(\mathcal{X}_{val}, w^*), \mathcal{Y}_{val})$$

$$w^* = \arg\min_{w} \mathcal{L}(f(\mathcal{T}_{\theta}(\mathcal{X}), w), \mathcal{Y})$$

Instead of learning a policy, we learn the parameters Θ of a network that generates stochastic augmentations

Before Θ was a limited set of configurations/policies → |Θ|<1000
Now Θ are the parameters of the Augmentation Network → |Θ|>>1000
We need to use gradient! Sampling approaches wouldn't work!

Automatic DA
Our approach



Uses an Augmentation Network with parameters Θ

Learns to generate transformations which reduce validation loss

Automatic DA
Approximations

$$\nabla_\theta \mathcal{L}(f(\mathcal{X}_{val}, w^*), \mathcal{Y}_{val})$$

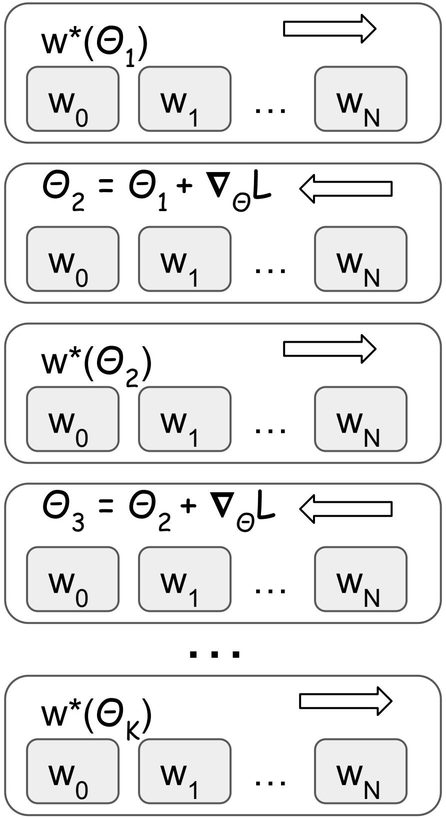
Approximate w* with one iteration of $\nabla_w \mathcal{L}(f(\mathcal{T}_\theta(\mathcal{X}), w), \mathcal{Y})$

$$\nabla_\theta \mathcal{L}(f(\mathcal{X}_{val}, w^*), \mathcal{Y}_{val}) \approx \nabla_\theta \mathcal{L}(f(\mathcal{X}_{val}, w - \nabla_w \mathcal{L}(\theta), \mathcal{Y}_{val})$$

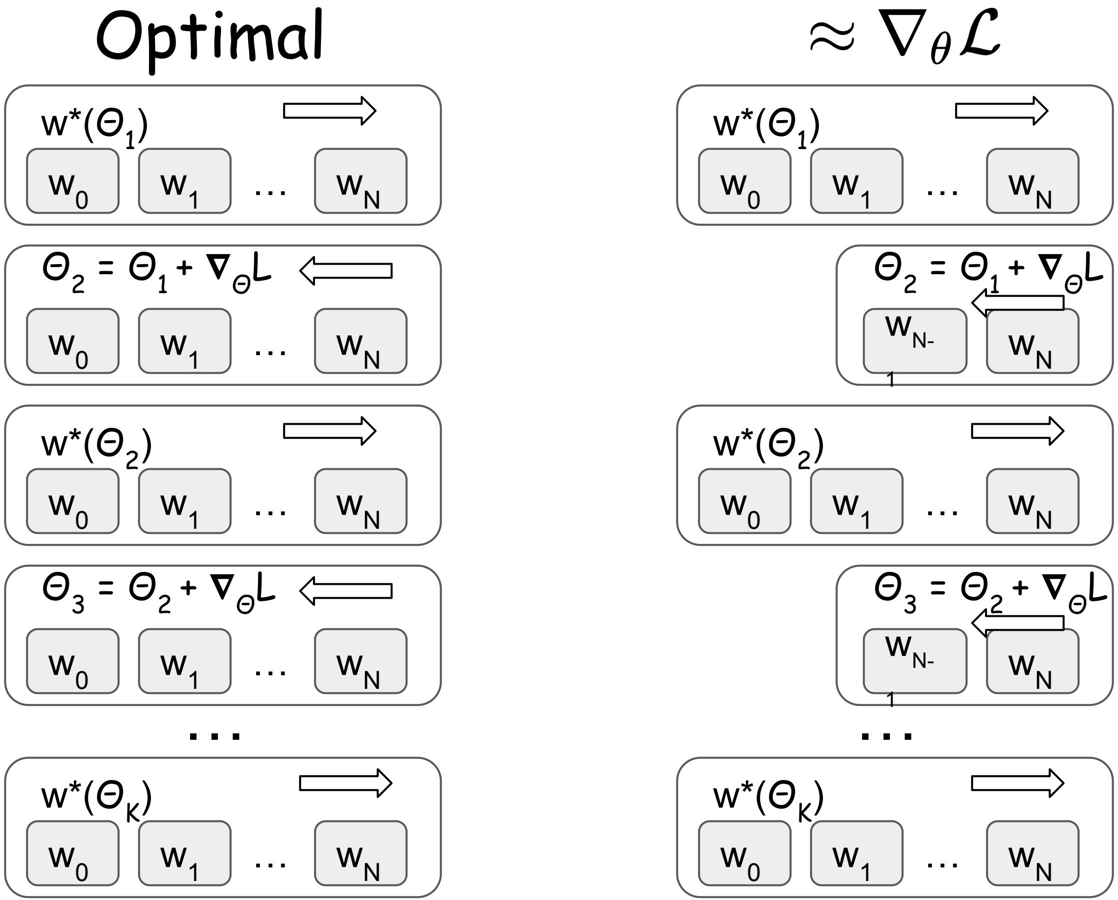
Approximate $\nabla_\theta \mathcal{L}$ with a single unroll of the gradient with respect ot w.

Automatic DA

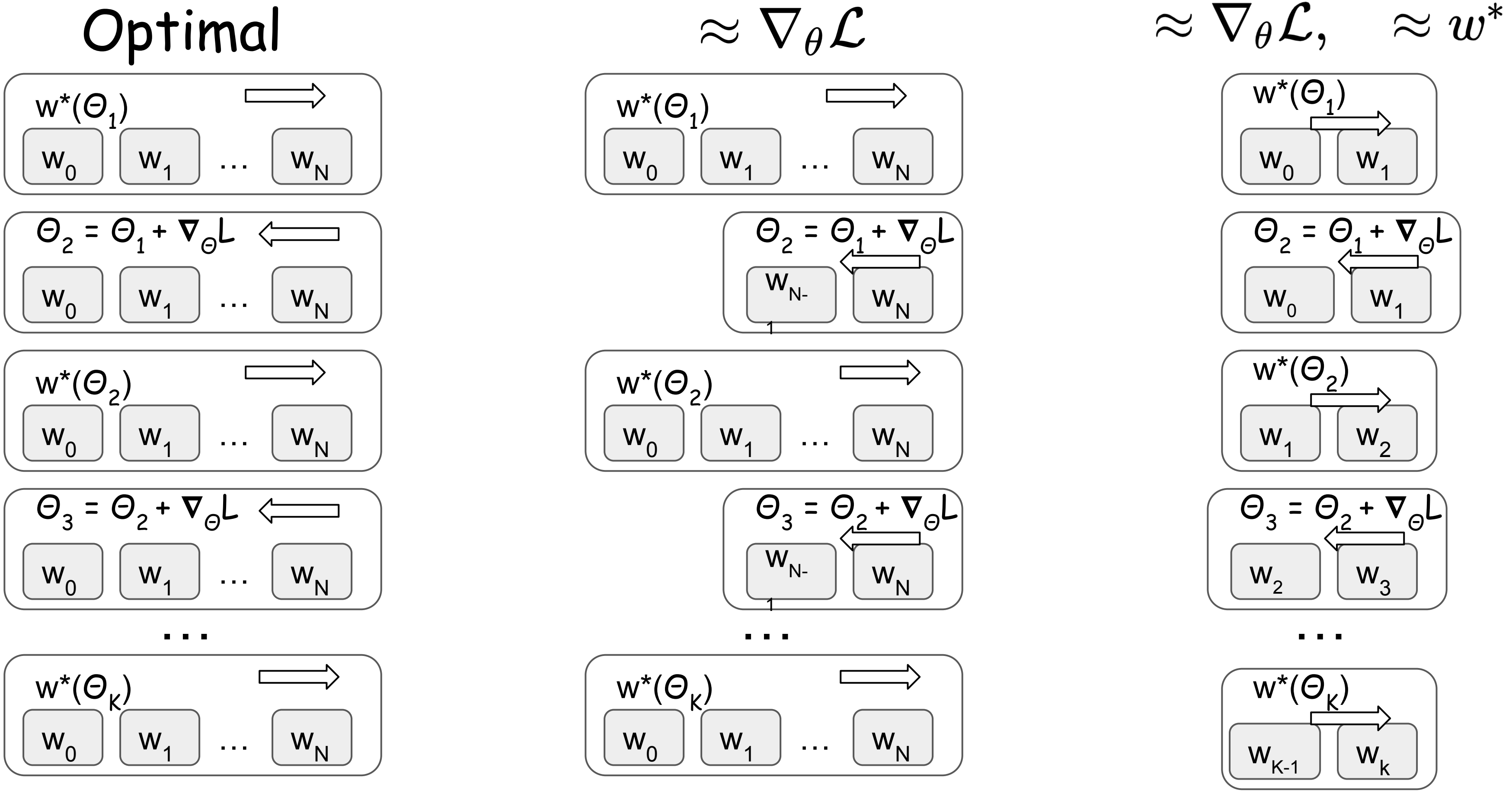
Optimal



Automatic DA



Automatic DA



Automatic DA
Results

ResNet18 / CIFAR10	**Trans.**	**Affine**	**Cost**
Baseline	88.55	88.55	1
Predefined	95.28	94.59	> 60
Transf. invariant (STN)	92.14	90.31	1.1
Validated magnitude	94.58	93.43	11.5
Our model	**95.35**	**95.16**	5.3

- Automatic DA is better than a transformation invariant model
- Augmentation Network is better than validated transformations

Automatic DA
Results

	Classifier	CIFAR10	CIFAR100
Baseline	ResNet18	88.55	68.99
Predefined	ResNet18	91.18	73.61
Bayesian DA [50]	ResNet18	91.00	72.10
DAN [36]	BadGAN	93.00	-
TANDA [42]	ResNet56	94.40	-
AutoAugment [9]	ResNet32	**95.50**	-
Ours	ResNet18	95.42	**74.31**
Baseline	WRN 28-10	94.83	69.90
Predefined	WRN 28-10	95.76	81.10
AutoAugment	WRN 28-10	**97.40**	82.90
Fast AA	WRN 28-10	97.30	82.70
PBA	WRN 28-10	**97.40**	**83.30**
RandAugment	WRN 28-10	97.30	**83.30**
Our model	WRN 28-10	96.44	81.90

- Comparable with more complex training approaches
- On larger models, policy based methods seems still better than our Augmenter based approach

Learning with Latent Variables

Masih Aminbeidokhti, Marco Pedersoli, Patrick Cardinal, Eric Granger, "Emotion recognition with spatial attention and temporal softmax pooling", ICIAR 2019 best paper award.

Théo Ayral, Marco Pedersoli, Simon Bacon, Eric Granger, "Temporal Stochastic Softmax for 3D CNNs: An Application in Facial Expression Recognition", WACV 2021.

Akhil Meethal, Marco Pedersoli, Zhing Zhu, Françisco P Romero, Eric Granger, "Semi-Weakly Supervised Object Detection by Sampling Pseudo Ground-Truth Boxes", IJCNN 2022.

Learning with LV

$$\mathcal{Y}^* = \arg\min_{\mathcal{Y}} \mathcal{L}(f(\mathcal{X}_{val}, w^*), \mathcal{Y}_{val})$$
$$w^* = \arg\min_{w} \mathcal{L}(f(\mathcal{X}, w), \mathcal{Y})$$

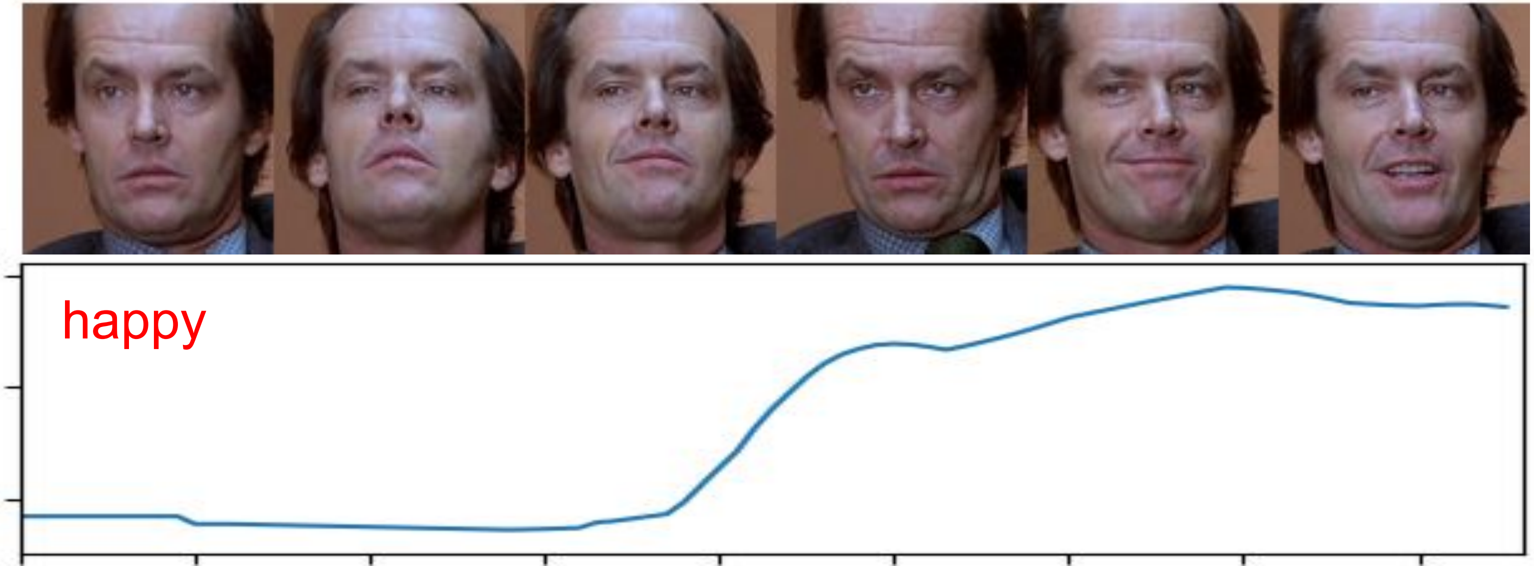
Simplified into:

$$\mathcal{Y}^* = \arg\min_{\mathcal{Y}} \mathcal{L}(f(\mathcal{X}, w^*), \mathcal{Y})$$
$$w^* = \arg\min_{w} \mathcal{L}(f(\mathcal{X}, w), \mathcal{Y})$$

Learning with LV

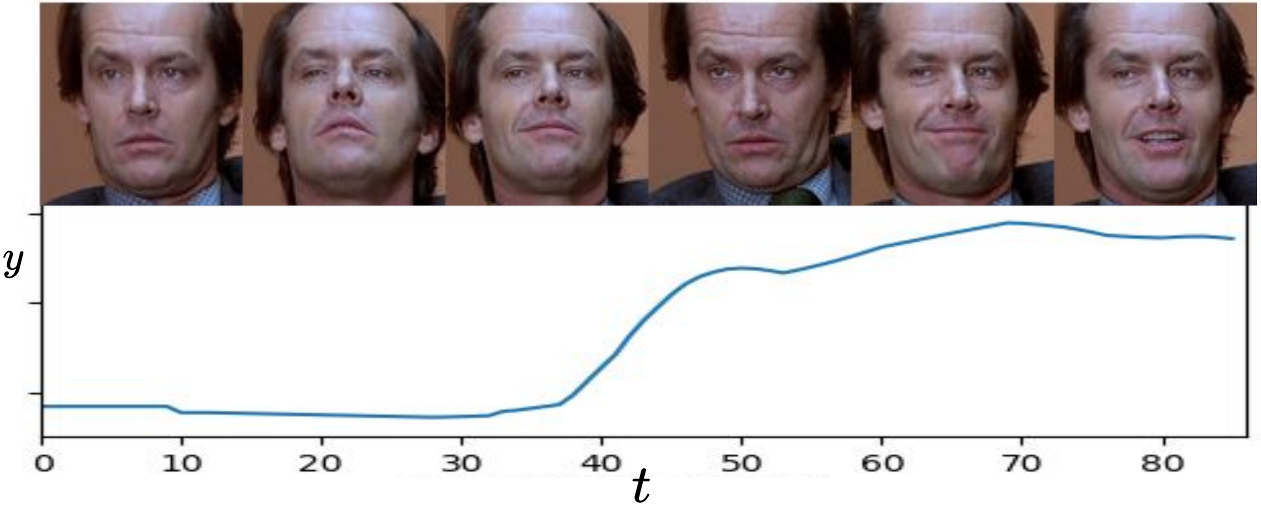
Temporal localization for FER

$$\mathcal{Y}^* = \arg\min_{\mathcal{Y}} \mathcal{L}(f(\mathcal{X}, w^*), \mathcal{Y}) \qquad w^* = \arg\min_{w} \mathcal{L}(f(\mathcal{X}, w), \mathcal{Y})$$



Temporal Localization

Learning with LV
Temporal localization for FER



$$f(x) = \sum_t y_t f(x_t)$$

$$y_t = \frac{\exp f(x_t)/T}{\sum_j \exp f(x_j)/T}$$

Softmax pooling: generalization of average and max pooling:

- when $T \to +inf$ average pooling
- when $T = 0$ max pooling

*But, large models **cannot fit the entire video** in memory!!!*

Learning with LV
Temporal localization for FER

Uniform Sampling of average pooling (previous approaches)*

$$f(x) = \frac{1}{N}\sum_{t=1}^{N} f(x_t) \quad \approx \frac{1}{K}\sum_{k=1}^{K} f(x_s)\,, \;\; s \sim \mathcal{U}$$

Uniform Sampling of weighted Temporal Pooling

$$f(x) = \sum_{t=1}^{N} y_t\, f(x_t) \quad \approx \frac{N}{K}\sum_{k=1}^{K} y_s\, f(x_s)\,, \;\; s \sim \mathcal{U}$$

Importance Sampling of weighted Temporal Pooling (ours)

$$f(x) = \sum_{t=1}^{N} y_t\, f(x_t) \quad \approx \frac{1}{K}\sum_{k=1}^{K} f(x_s)\,, \;\; s \sim \mathcal{M}(y)$$

* *Joao Carreira, Andrew Zisserman,"Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset", CVPR 2017.*

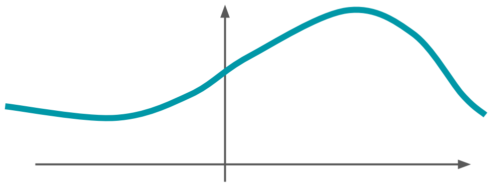
Learning with LV
Temporal localization for FER

Uniform Sampling of average pooling

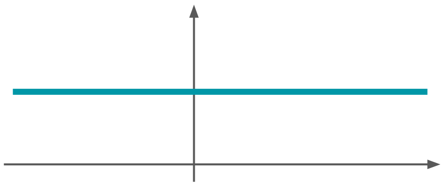
$$f(x) = \frac{1}{N}\sum_{t=1}^{N} f(x_t) \quad \approx \frac{1}{K}\sum_{k=1}^{K} f(x_s)\ , \quad s \sim \mathcal{U}$$

+ Reduced memory and computation
+ Same objective in expectation
- Considering every part of the video in the same way
- Increased variance due to the sum estimation

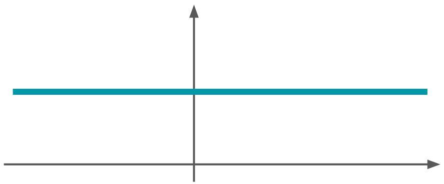
Function



Pooling



Sampling



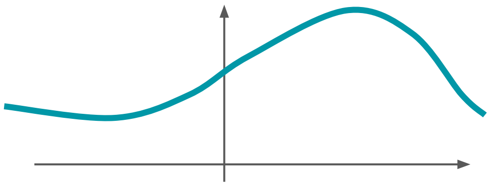
Learning with LV
Temporal localization for FER

Uniform Sampling of weighted Temporal Pooling

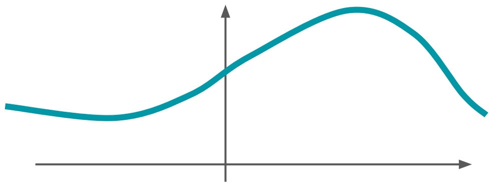
$$f(x) = \sum_{t=1}^{N} y_t f(x_t) \quad \approx \frac{N}{K} \sum_{k=1}^{K} y_s f(x_s)\,, \quad s \sim \mathcal{U}$$

+ Reduced memory and computation
+ Same objective in expectation
+ Can focus on the most important frames with Softmax Pooling
+ Can still estimate w with backprop as using uniform sampling
- High variance due to the uniform sampling

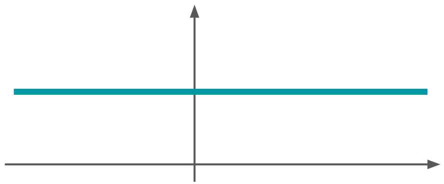
Function



Pooling



Sampling

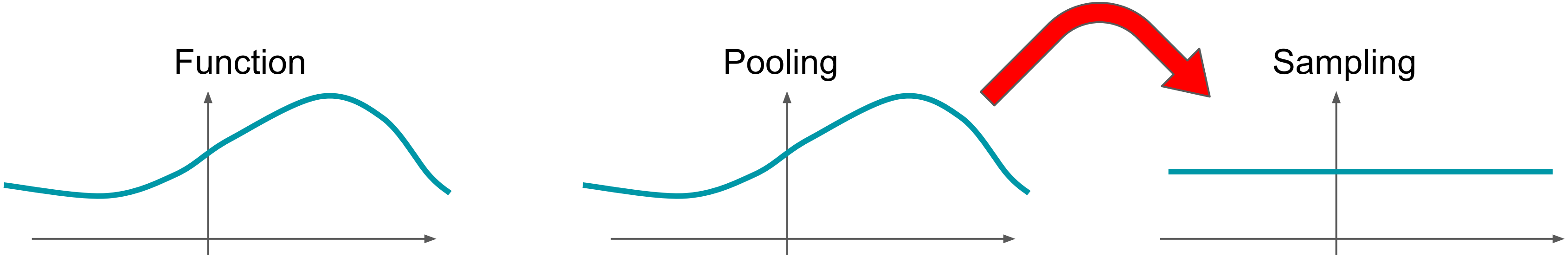


Learning with LV
Temporal localization for FER

Importance Sampling of weighted Temporal Pooling

$$f(x) = \sum_{t=1}^{N} y_t f(x_t) \quad \approx \frac{N}{K} \sum_{k=1}^{K} y_s f(x_s)\,, \;\; s \sim \mathcal{U}$$

Instead of applying a weight y to f
we sample with an importance y
Same objective but lower variance!

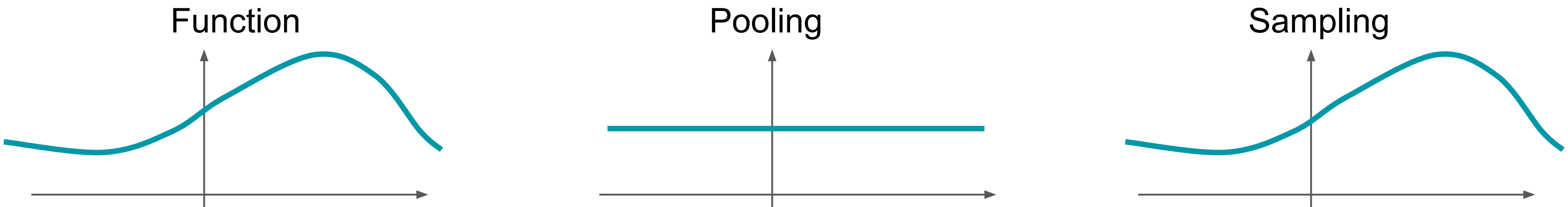


Learning with LV
Temporal localization for FER

Importance Sampling of weighted Temporal Pooling

$$f(x) = \sum_{t=1}^{N} y_t f(x_t) \quad \approx \frac{1}{K} \sum_{k=1}^{K} f(x_s)\,, \;\; s \sim \mathcal{M}(y)$$

+ Reduced memory and computation
+ Same objective in expectation
+ Can focus on the most important frames with Softmax Pooling
+ Reduced variance due to the Multinomial sampling proportional to the frame importance
- Cannot estimate w with backpropagation due to Multinomial sampling



Learning with LV
Our Approach

Multinomial Sampling proportional to y

$$f(x) = \sum_{t=1}^{N} y_t f(x_t) \qquad \approx \frac{1}{K} \sum_{k=1}^{K} f(x_s)\,, \quad s \sim \mathcal{M}(y)$$

How to estimate y?

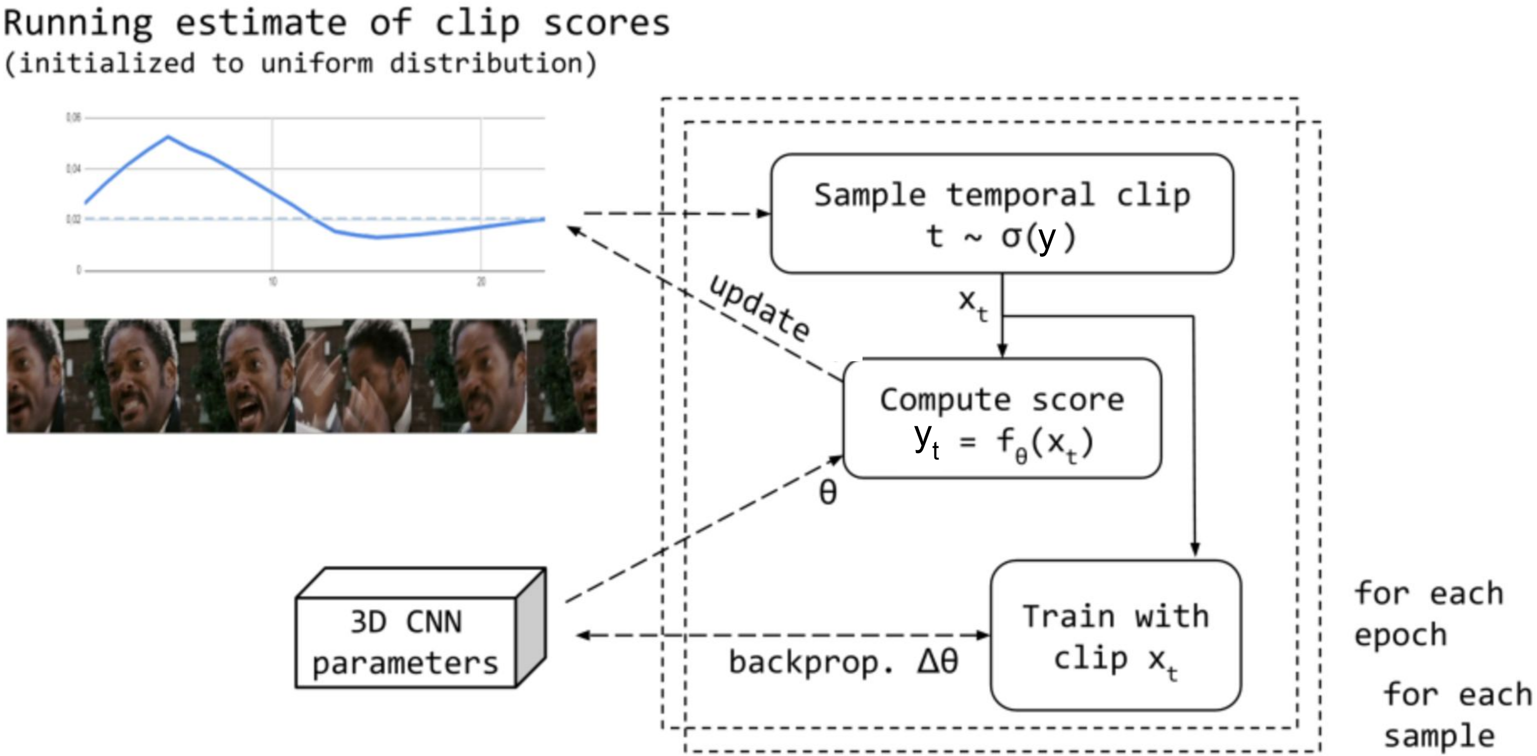
For each sample s: $q_s = \beta q_s + (1-\beta) f(x_s)$

Re-normalize: $y_t = \dfrac{\exp(q_t)}{\sum_i \exp(q_i)}$

Thus: $y_t \approx \dfrac{\exp(f(x_s))}{\sum_i \exp(f(x_i))}$

Learning with LV

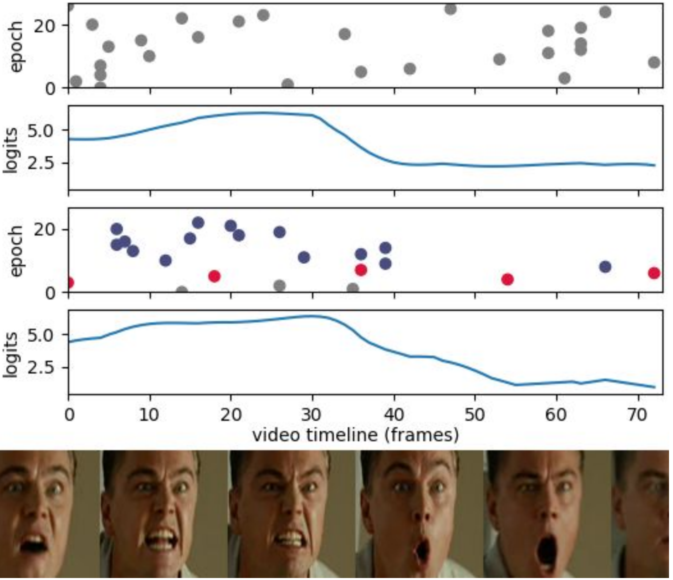
Our Approach

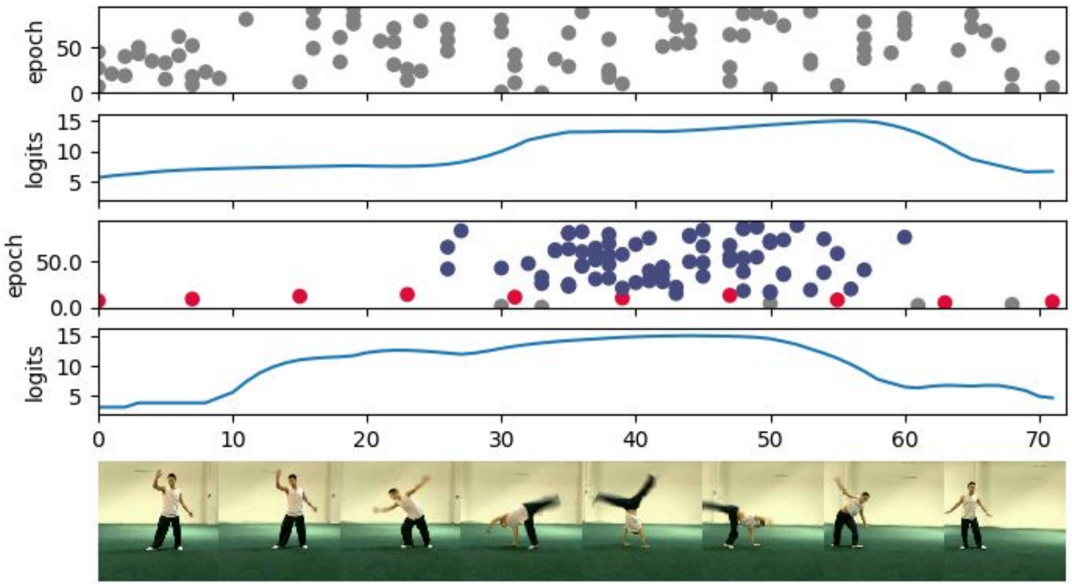


Learning with LV
Results

Uniform Sampling

Weighted Sampling





Learning with LV
Results

Inverse Temp.	REINFORCE		Ours Softmax	
	Acc.(%)	Ep.	Acc.(%)	Ep.
$\gamma = 0$	45.66 ±.21	24.6	45.66 ±.21	24.6
$\gamma = 0.5$	46.09 ±.41	23.8	46.07 ±.27	23.6
$\gamma = 1$	46.80 ±.63	22.5	**47.35** ±.27	20.3
$\gamma = 10$	44.52 ±.18	17.5	46.65 ±.40	17.2

- With correct temperature faster and better training than uniform sampling

Learning with LV
Results

Method	Model	Acc. (%)
Lu *et al.*, 2018 [31]	3D VGG-16	39.36
Fan *et al.*, 2016 [11]	C3D	39.69
Vielzeuf *et al.*, 2017 [51]	C3D-LSTM	43.2
	C3D Weighted	42.1
C3D baseline	C3D (uniform)	39.95
C3D with Softmax	C3D ($\gamma = 1$)	42.78
VGG baseline	3D VGG-16 (unif.)	45.66
VGG with Softmax	3D VGG-16 ($\gamma = 1$)	47.35

- Better than other models with uniform sampling on different backbones

Conclusions

- In learning, we need not only to fit the data, but also to:
  - generalize
  - select the model
  - learn with missing/noisy data

- We can cast all these problems as bilevel optimization

- Thus, **learning is a Bilevel Optimization problem**!

Looking for collaborations

- **Bilevel optimization**
  - Better theoretical understanding
  - New approximations and applications
- **Sampling-based learning**
  - Connections with RL and multi-armed Bandit
  - Further exploration / new applications
- **Transformer/Large Language Models**
  - Reduce quadratic constraints
  - Work on connection between text and vision