Transfer and Multi-task Learning

AI602: Recent Advances in Deep Learning

Lecture 5

Slide made by

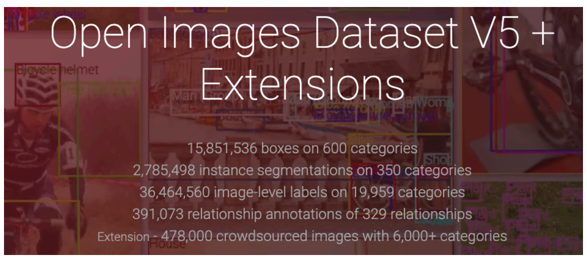
Jongjin Park, Yunhun Jang, Hankook Lee and Jihoon Tack

KAIST EE

Table of Contents

Table of Contents

- **Deep learning suffers from a lack of training samples**
  - Deep learning shows remarkable success in various fields of artificial intelligence (e.g., object classification, machine translation)
  - But, use (VERY) large labeled dataset



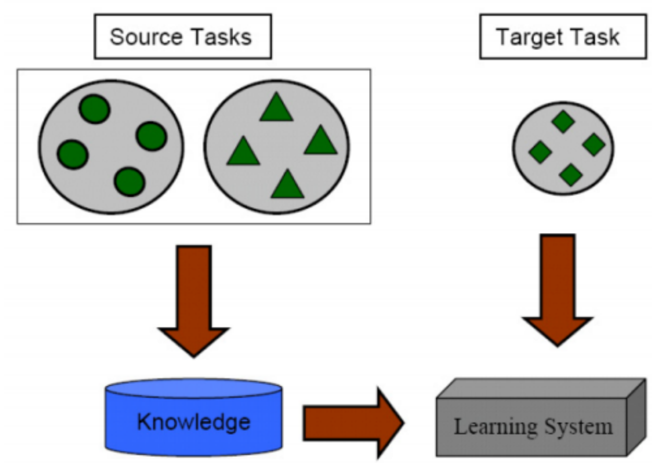
Open Images Dataset (9M images)

English Wikipedia (2.5B words)

>50 bounding boxes in an image

  - **Collecting some annotations is too hard/expensive**
    - E.g., segmentation labels, bounding boxes, medical data
    - For a new task, only few samples are available

- ***Transfer learning*** aims to **transfer the knowledge** from **source** to **target domains & tasks**

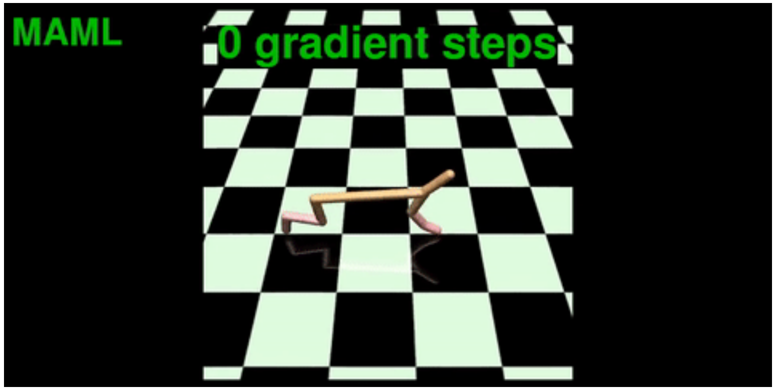


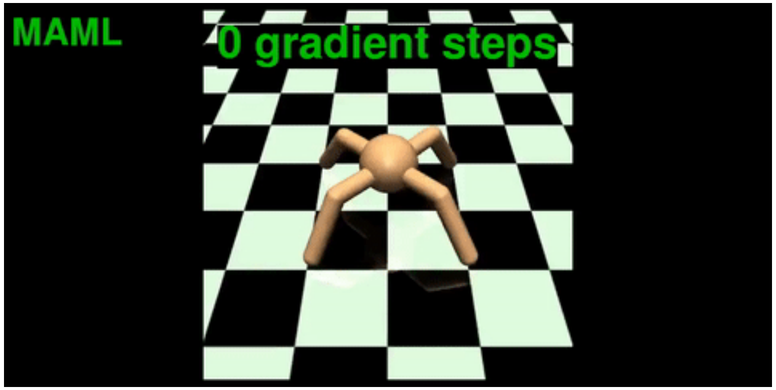
* source: https://storage.googleapis.com/openimages/web/index.html , [Pan et al., 2010]

Transfer Learning in Artificial Intelligence

Robots learns skills and transfers that knowledge to other robots have different kinematics

Speech recognition: Learn from specific languages/accents transfer to learn different languages/accents



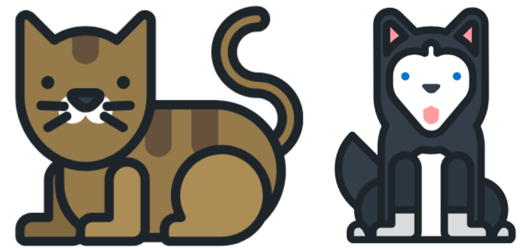


Simulated robots learn new movements from get transfer from previous learned task
(Top): from forward movements, learn backward move
(Bottom): learn faster movements from slow movements

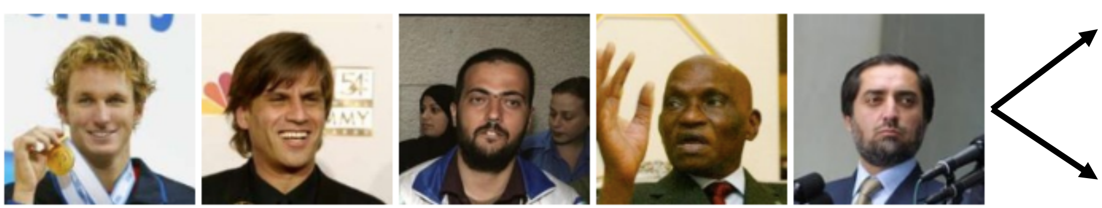
* source: https://www.youtube.com/watch?v=_t1c_IrEH1k

- **Domain** $\mathcal{D} = \{\mathcal{X}, P(X)\}$
  - With a feature space $\mathcal{X}$ and a marginal probability distribution $P(X)$ for $X \in \mathcal{X}$
  - E.g., $\mathcal{X}$ is natural or cartoon image spaces / $P(X)$ is dog or cat distribution





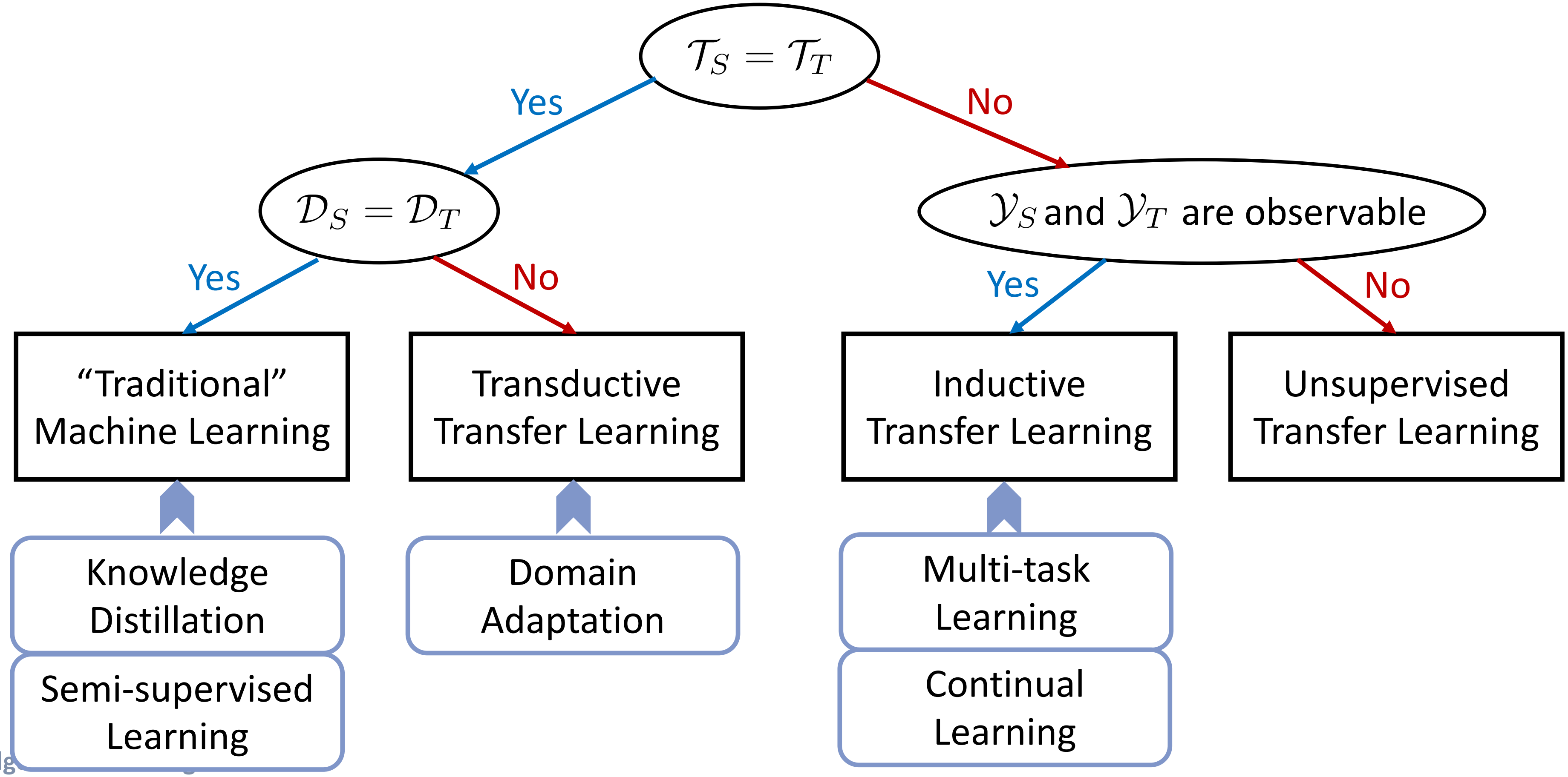
- **Task** $\mathcal{T} = \{\mathcal{Y}, P(Y|X)\}$
  - With a label space $\mathcal{Y}$ and a conditional probability distribution $P(Y|X)$ for $Y \in \mathcal{Y}$
  - E.g., $\mathcal{Y}$ is digit (0, 1, …) or animal (dog, cat, …) spaces



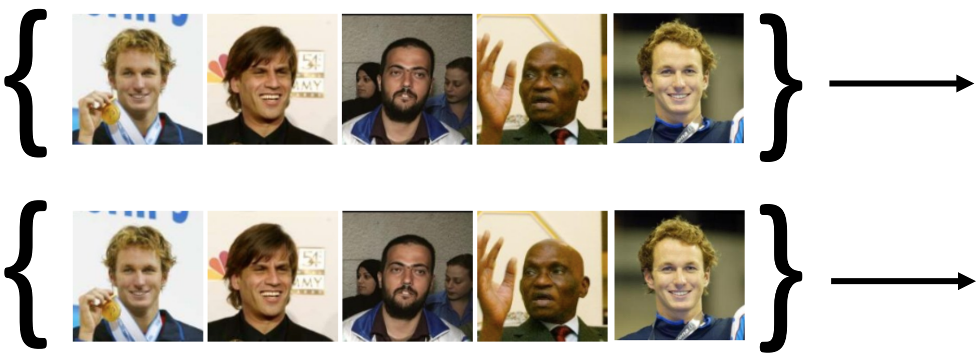
Age (e.g., 31, 49, 34, 50, 31)

Person recognition
(e.g., John, Aaron, Adam, Will, John)

- Definition of transfer learning [Pan et al., 2010]
  - Given a source domain $\mathcal{D}_S$ and learning task $\mathcal{T}_S$, and a target domain $\mathcal{D}_T$ and learning task $\mathcal{T}_T$
  - ***Transfer learning*** aims to **improve** the learning of **the target predictive function** $f_T(\cdot)$ using the knowledge in $\mathcal{D}_S$ and $\mathcal{T}_S$ where $\mathcal{D}_S \neq \mathcal{D}_T$ or $\mathcal{T}_S \neq \mathcal{T}_T$



- When tasks and domains are same, usually one can transfer knowledge for
  - Making target model that are smaller (model compression)
  - But, perform better than scratch learning
  - Using the knowledge transferred from the source model
- Knowledge distillation
  - Make a target model mimic the source model
    - Make outputs (or features) similar
    - Since tasks and domains are same, following a source/reference model is useful



Person recognition
(e.g., John, Aaron, Adam, Will, John)

Person recognition
(e.g., John, Aaron, Adam, Will, John)

$\mathcal{D}_S = \mathcal{D}_T$ $\mathcal{T}_S = \mathcal{T}_T$

Type II: Same Tasks, but Different Domains (Transductive Transfer Learning)

- Labels to predict are same but input data samples are different
  - Since tasks are same, by learning the features *invariant* to source and target domains, a target model can perform well
  - In many cases, target domain datasets do not have sufficient labels
    - By learning domain invariant features, source model's representations could be used for target domain
  - ***Domain adaptation*** (not covered in this lecture)
    - Learn representations that confuse source and target domain inputs
    - Learn target representations that are similar to source domain

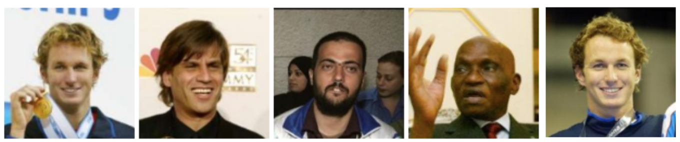
0-9 digits classification
(e.g., 6, 2, 3, 8, 1, 0)

0-9 digits classification
(e.g., 0, 4, 8, 5, 3, 9)

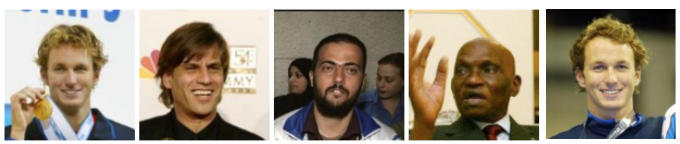
$\mathcal{D}_S \neq \mathcal{D}_T$ $\quad$ $\mathcal{T}_S = \mathcal{T}_T$

Type III: Different Tasks (Inductive/Unsupervised Transfer Learning)

- Different tasks: different labels to predict
  - When tasks are different, feature extractors and output layers are need to be adjusted a lot for new tasks
  - Multi-task learning/fine-tuning are used to learn appropriate representations for target tasks from the source model's representations



Age (e.g., 31, 49, 34, 50, 31)

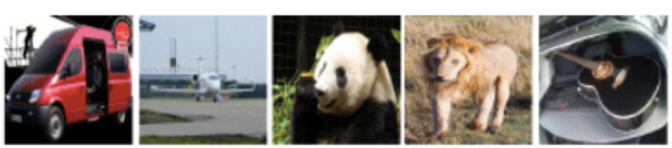


Person recognition
(e.g., John, Aaron, Adam, Will, John)

$\mathcal{D}_S = \mathcal{D}_T$

$\mathcal{T}_S \neq \mathcal{T}_T$

Scene classification
(e.g., elevator, gas station, castle, cafeteria, cabin)



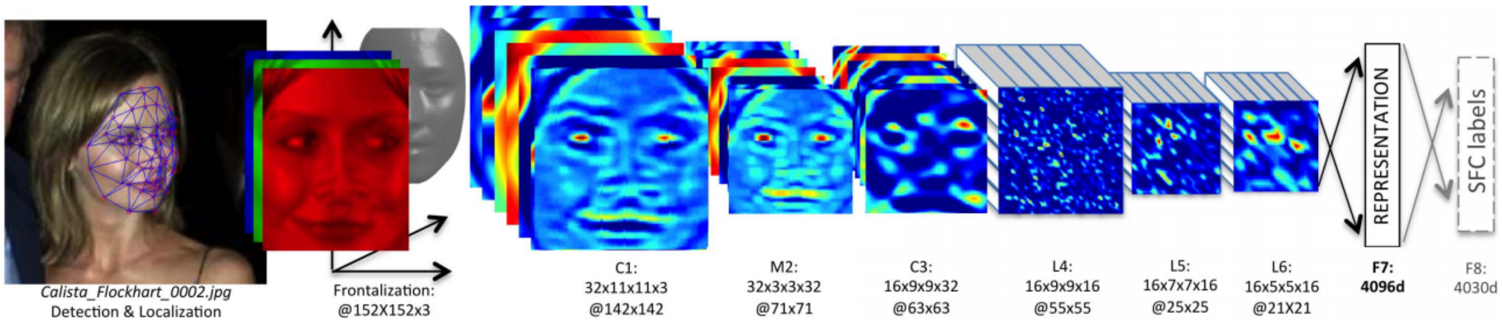
Object classification
(e.g., car, airplane, panda, lion, guitar)

$\mathcal{D}_S \neq \mathcal{D}_T$

$\mathcal{T}_S \neq \mathcal{T}_T$

Table of Contents

- Convolutional layers are viewed as a feature extractor.
  - Lower convolutional layers capture low-level features. e.g. edges
  - Higher convolutional layers capture more complex, high-level features. e.g. eyes



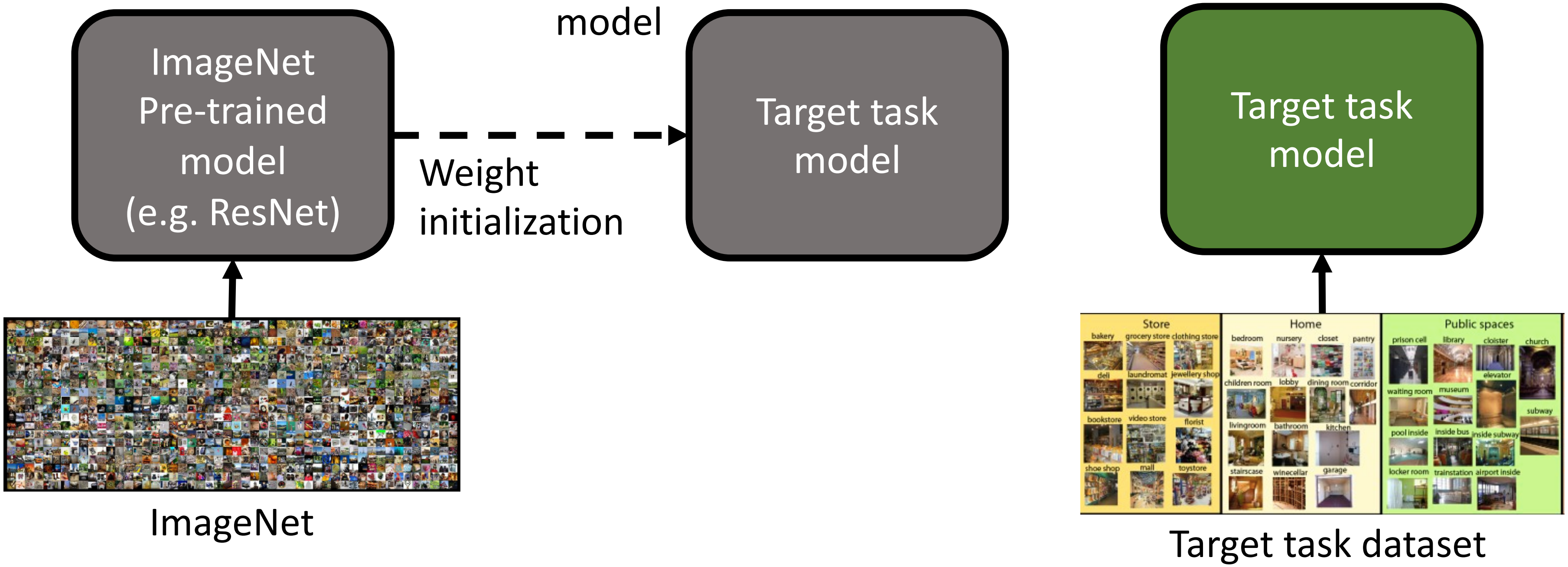
- A source model pre-trained by a large dataset, e.g., ImageNet, is well-generalized, so one can expect it as a *good feature extractor or parameter initialization*.
  - To avoid overfitting, one can often *freeze* convolutional layers for small target datasets.
  - Can transfer to different domains and tasks
  - But, same architectures (at least for feature extraction part)

* source: [Yaniv et al., 2014]

(a) Train large model on ImageNet

(b) Using pre-trained weights as initial parameter of target model

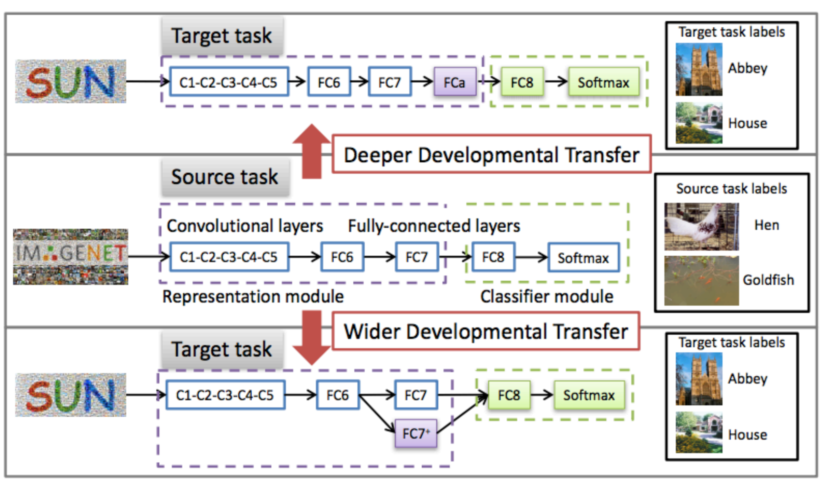
(c) Fine-tuning the weights for new task

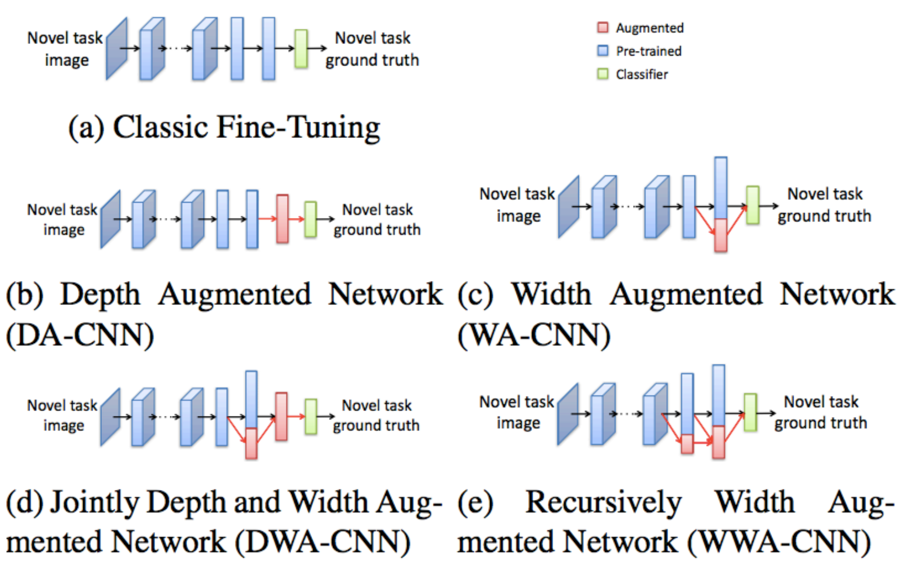


- Assumptions for fine-tuning approaches
  - **Features/Parameters** learned from some task are useful for **another tasks**
    - True in many artificial intelligence tasks (e.g. lower-level features of images such as edge)
- When do they **fail** to work
  - When **dataset** of source and target tasks are very **different**
  - When target tasks **have no (or very small) labeled training data**

Fine-Tuning with Increasing Target Model Capacity

- Increasing the target model capacity in various ways [Wang et al., 2017]
  - Channel-wise, depth-wise, (channel+depth)-wise
  - Using the pre-trained weights for all the layers except newly augmented layers/channels
  - Fine-tuning with target tasks
- Main idea at a high level
  - Using the pre-trained weight of source model to initialize the target model
  - Increase the capacity of target model in depth/channel-wise





* source: [Wang et al., 2017]

Experimental Results

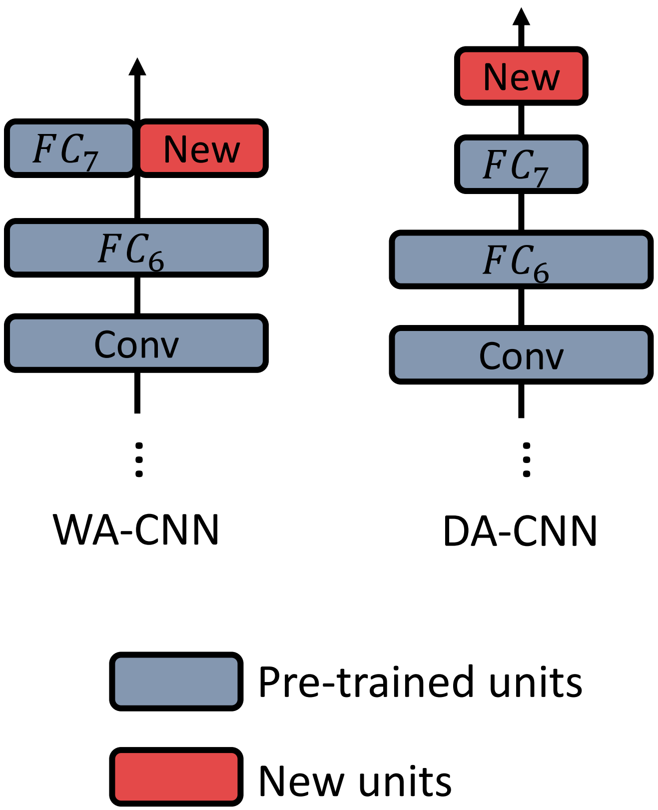
- Evaluated on MIT-67, 102 Flowers, CUB200-2011, Stanford-40 with ImageNet pre-trained AlexNet
- Outperform most of task customized CNN or other multi-task learning methods
- Drawbacks:
  - Did not apply on architecture like ResNet (model without fully-connected layers)
  - Only augment the layers for fully-connected layers

Type	MIT-67		102 Flowers		CUB200-2011		Stanford-40	
	Approach	Acc(%)	Approach	Acc(%)	Approach	Acc(%)	Approach	Acc(%)
ImageNet CNNs	Finetuning-CNN	61.2	Finetuning-CNN	75.3	Finetuning-CNN	62.9	Finetuning-CNN	57.7
	Caffe [53]	59.5	CNN-SVM [32]	74.7	CNN-SVM [32]	53.3	Deep Standard [4]	58.9
	—	—	CNNaug-SVM [32]	86.8	CNNaug-SVM [32]	61.8	—	—
Task Customized CNNs	Caffe-DAG [53]	64.6	LSVM [30]	87.1	LSVM [30]	61.4	Deep Optimized [4]	66.4
	—	—	MsML+ [30]	89.5	DeCaf+DPD [7]	65.0	—	—
	Places-CNN [59]	**68.2**	MPP [55]	91.3	MsML+ [30]	66.6	—	—
	—	—	Deep Optimized [4]	91.3	MsML+* [30]	67.9	—	—
Data Augmented CNNs	Combined-AlexNet [18]	58.8	Combined-AlexNet [18]	83.3	—	—	Combined-AlexNet [18]	56.4
Multi-Task CNNs	Joint [22]	63.9	—	—	Joint [22]	56.6	—	—
	LwF [22]	64.5	—	—	LwF [22]	57.7	—	—
Ours	WA-CNN	66.3	WA-CNN	**92.8**	WA-CNN	**69.0**	WA-CNN	**67.5**

- Normalization and scaling activations are important for the performance improvement
  - Reconcile the learning pace of the new and pre-existing units
  - Normalization and scaling is more crucial in Width-augmented CNN (WA-CNN)
    - Without normalization and scaling, marginally better or worse than fine-tuning method

$$\hat{\boldsymbol{h}}^k = \gamma \boldsymbol{h}^k / \left\|\boldsymbol{h}^k\right\|_2$$

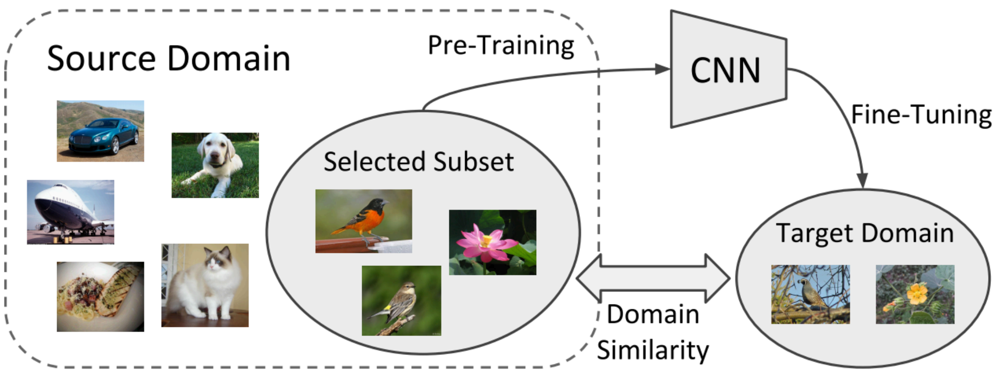
Scaling (γ)   Normalization ($/ \left\|\boldsymbol{h}^k\right\|_2$)

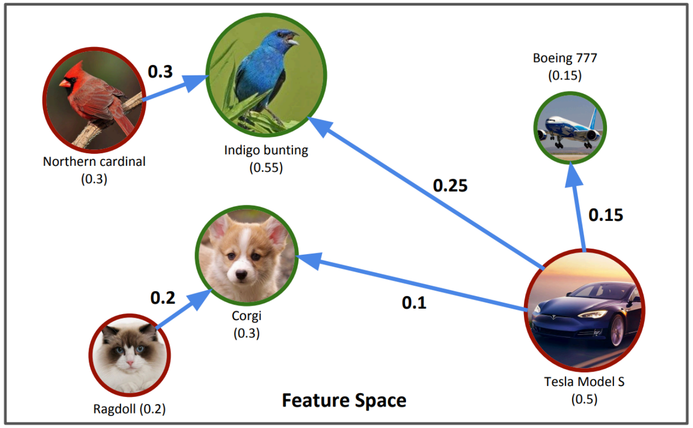


Method	Scaling	New	FC_7-new	FC_6-new	All
Fine-tuning CNN	-	53.63	54.75	54.29	55.93
DA-CNN	w/o (rand)	**53.82**	**56.47**	56.25	57.21
	w/	53.51	56.15	**57.14**	**58.07**
WA-CNN	w/o (rand)	53.78	54.66	49.72	51.34
	w/o (copy+rand)	53.62	54.35	53.70	55.31
	w/	**56.81**	**56.99**	**57.84**	**58.95**

Performance on SUN-397 dataset by changing the fine-tuning layers from only new layer to all the layers
w/o (rand): new units are randomly initialized
w/o (copy+rand): initialize by copying FC_7, and add random noise
w/: with normalization and scaling

- **Select samples** from the source domain before pre-training [Cui et al., 2018]
  - Select ***only related samples*** to the target task
  - Pre-train CNNs on the selected set $\Longrightarrow$ Fine-tune CNNs on the target dataset





- *How to measure similarity* between samples in source/target datasets?
  - $d(s,t) = \|g(s) - g(t)\|$ for $s \in \mathcal{X}_S, t \in \mathcal{X}_T$ where $g(\cdot)$ is a feature extractor
  - **Earth Mover's Distance**: minimum cost of moving samples between two sets

$$d(\mathcal{S},\mathcal{T}) = \mathrm{EMD}(\mathcal{S},\mathcal{T}) = \min_f \frac{\sum_{i,j} f_{i,j} d(s_i, t_j)}{\sum_{i,j} f_{i,j}}$$

    - With some constraints
  - Incrementally select samples in the source domain using a greedy strategy

* source: [Cui et al., 2018]

- Transfer learning performance on the selected subsets

target

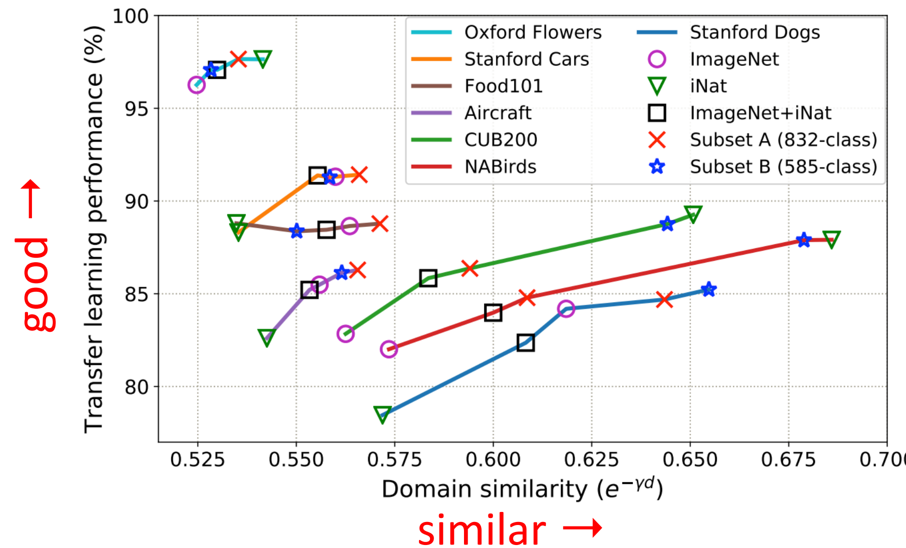
source

	CUB200	Stanford Dogs	Flowers-102	Stanford Cars	Aircraft	Food101	NABirds
ImageNet	82.84	84.19	96.26	91.31	85.49	88.65	82.01
iNat	89.26	78.46	97.64	88.31	82.61	88.80	87.91
ImageNet + iNat	85.84	82.36	97.07	91.38	85.21	88.45	83.98
Subset A (832-class)	86.37	84.69	97.65	91.42	86.28	88.78	84.79
Subset B (585-class)	88.76	85.23	97.37	90.58	86.13	88.37	87.89

  - Pre-training on selected subsets achieves good performance consistently

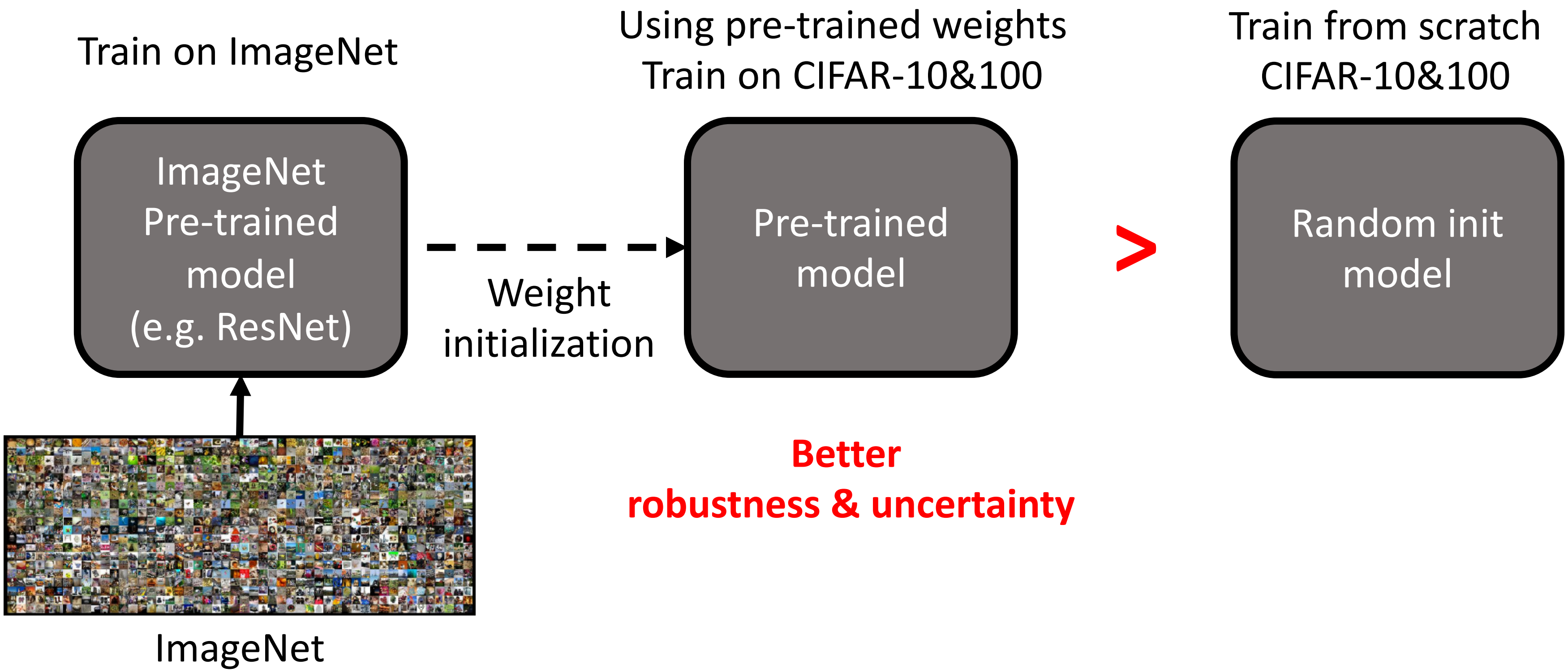
- The relationship between transfer learning performance and domain similarity

- **Drawbacks**
  - Pre-training for each model
    - Pre-training takes few hours ~ days
  - Need all samples in source domain
    - # of samples in source datasets > 1M



* source: [Cui et al., 2018]

- Pre-training also improves other tasks such as **robustness and uncertainty**
- Considered various scenarios such as **label corruption, class imbalance, out-of-distribution detection**, etc.



- **Label corruption:** when mis-labeled sample existed in train data

	CIFAR-10		CIFAR-100	
	Normal Training	Pre-Training	Normal Training	Pre-Training
No Correction	28.7	15.9	55.4	39.1
Forward Correction	25.5	15.7	52.6	42.8
GLC (5% Trusted)	14.0	7.2	46.8	33.7
GLC (10% Trusted)	11.5	6.4	38.9	28.4

- **Class imbalance**: when labels are imbalanced

Dataset	Method \ Imbalance Ratio	0.2	0.4	0.6	0.8	1.0	1.5	2.0
		Total Test Error Rate / Minority Test Error Rate (%)						
CIFAR-10	Normal Training	23.7 / 26.0	21.8 / 26.5	21.1 / 25.8	20.3 / 24.7	20.0 / 24.5	18.3 / 23.1	15.8 / 20.2
	Cost Sensitive	22.6 / 24.9	21.8 / 26.2	21.1 / 25.7	20.2 / 24.3	20.2 / 24.6	18.1 / 22.9	16.0 / 20.1
	Oversampling	21.0 / 23.1	19.4 / 23.6	19.0 / 23.2	18.2 / 22.2	18.3 / 22.4	17.3 / 22.2	15.3 / 19.8
	SMOTE	19.7 / 21.7	19.7 / 24.0	19.2 / 23.4	19.2 / 23.4	18.1 / 22.1	17.2 / 22.1	15.7 / 20.4
	Pre-Training	8.0 / 8.8	7.9 / 9.5	7.6 / 9.2	8.0 / 9.7	7.4 / 9.1	7.4 / 9.5	7.2 / 9.4
CIFAR-100	Normal Training	69.7 / 72.0	66.6 / 70.5	63.2 / 69.2	58.7 / 65.1	57.2 / 64.4	50.2 / 59.7	47.0 / 57.1
	Cost Sensitive	67.6 / 70.6	66.5 / 70.4	62.2 / 68.1	60.5 / 66.9	57.1 / 64.0	50.6 / 59.6	46.5 / 56.7
	Oversampling	62.4 / 66.2	59.7 / 63.8	59.2 / 65.5	55.3 / 61.7	54.6 / 62.2	49.4 / 59.0	46.6 / 56.9
	SMOTE	57.4 / 61.0	56.2 / 60.3	54.4 / 60.2	52.8 / 59.7	51.3 / 58.4	48.5 / 57.9	45.8 / 56.3
	Pre-Training	37.8 / 41.8	36.9 / 41.3	36.2 / 41.7	36.4 / 42.3	34.9 / 41.5	34.0 / 41.9	33.5 / 42.2

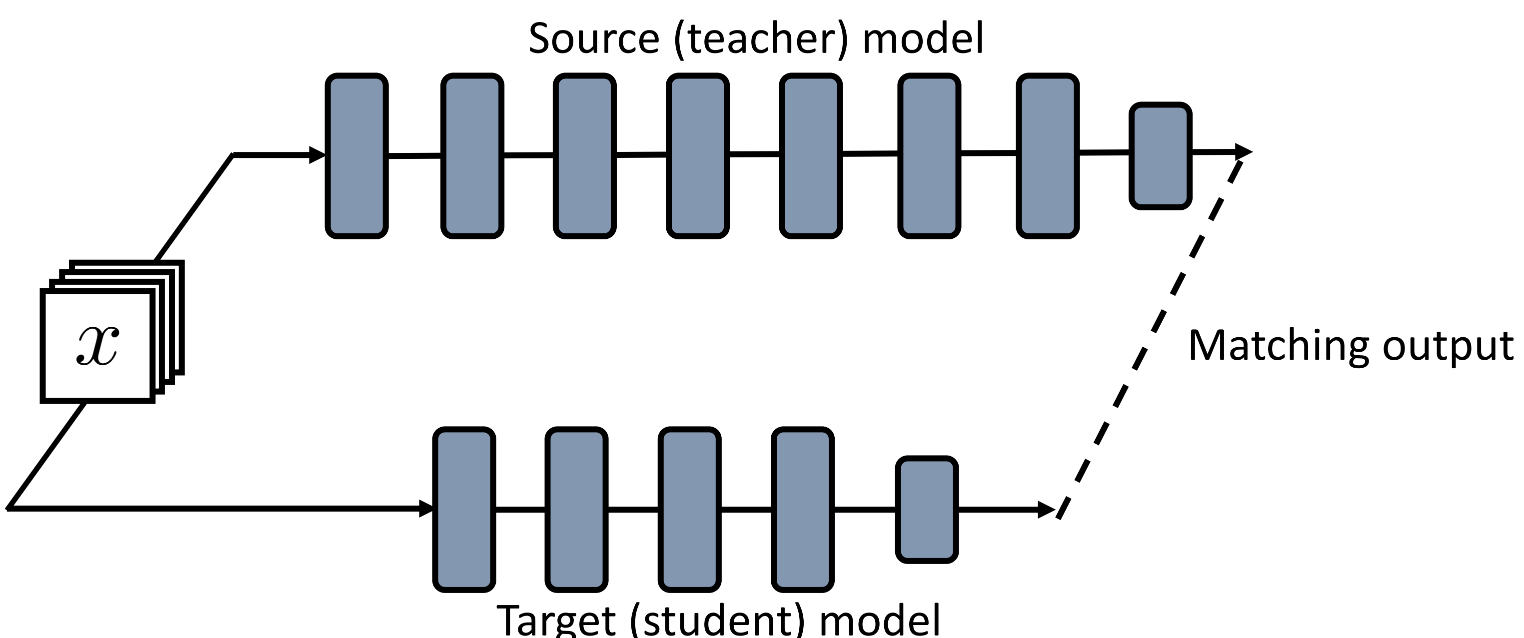
- **Out-of-distribution detection**: detecting unseen samples in the test set

	AUROC		AUPR	
	Normal	Pre-Train	Normal	Pre-Train
CIFAR-10	91.5	94.5	63.4	73.5
CIFAR-100	69.4	83.1	29.7	52.7
Tiny ImageNet	71.8	73.9	30.8	31.0

* source: [Hendrycks et al., 2019]

Table of Contents

- Learn a source model and distill its knowledge to a target model
  - Can lead to a better model with small architecture, or faster training
- Given a teacher network on domain $\mathcal{D}$, enhance the training of (usually **smaller**) a student network on ***same*** domain $\mathcal{D}$, using knowledge of a teacher network
- Done by **matching the output** of source and target models
  - Design a new loss term (e.g., MSE loss, KL divergence) for making source and target outputs similar in addition to the original loss term (e.g., cross entropy loss)



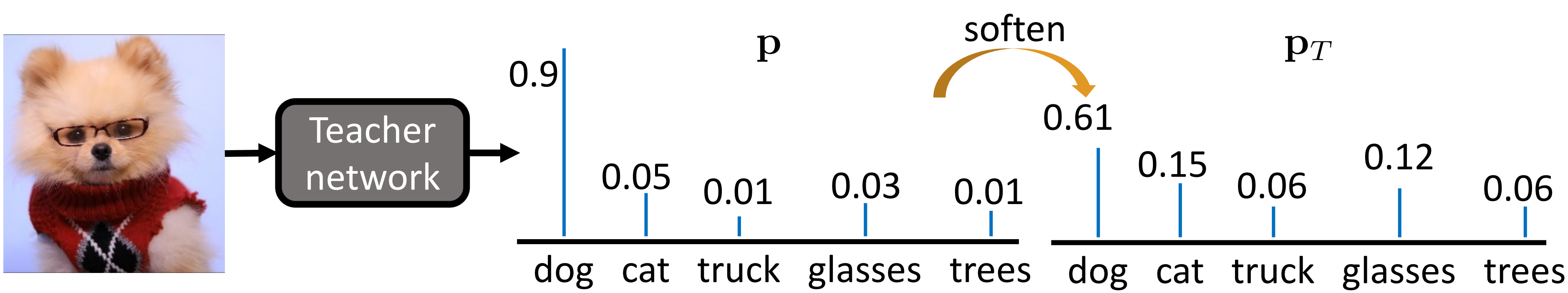
- [Hinton et al., 2015] propose
  - Use temperature $T \geq 1$ to make a *softer* probability distribution over classes

$$q_{i,T} = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)}$$

  where z_i, q_i are *the* i-th logit and probability, respectively
  - Use the *soft target* as additional labels to train student model

$$\mathcal{L} = (1-\alpha)\mathcal{L}_{\text{ce}}(\mathbf{y}, \mathbf{q}) + \alpha T^2 \mathcal{L}_{\text{ce}}(\mathbf{p}_T, \mathbf{q}_T)$$

where $\mathbf{y}$, $\mathbf{q}$ and $\mathbf{p}$ are ground-truth labels, target model outputs, and source model outputs, respectively. It is important to **multiply soft targets by T^2** because the magnitudes of the gradients produced by them scale as $1/T^2$. (derived in the next page)



Knowledge Distillation: Effect of Temperature Scaling

- Let C be a cross-entropy loss of softened labels.

$$C = \mathcal{L}_{\text{ce}}(\mathbf{p}_T, \mathbf{q}_T)$$

- The gradient of C, with respect to each target logit z_i, and source logit v_i :

$$\frac{\partial C}{\partial z_i} = \frac{1}{T}(q_i - p_i) = \frac{1}{T}\left(\frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)}) - \frac{\exp(v_i/T)}{\sum_j \exp(v_j/T)}\right)$$

  - If the temperature is high compared with the magnitude of the logits,

$$\frac{\partial C}{\partial z_i} \approx \frac{1}{T}\left(\frac{1+z_i/T}{N+\sum_j z_j/T} - \frac{1+v_i/T}{N+\sum_j v_j/T}\right)$$

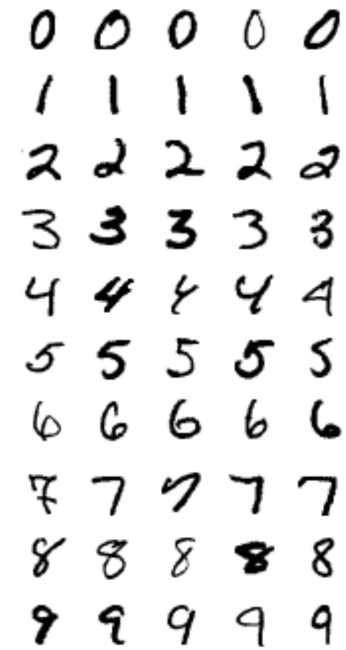
  - If we assume that the logits have been zero-meaned (i.e. $\sum_j z_j = \sum_j v_j = 0$)

$$\frac{\partial C}{\partial z_i} \approx \frac{1}{NT^2}(z_i - v_i) = \boxed{\frac{1}{NT^2}} \frac{\partial}{\partial z_i}\left(\boxed{\frac{1}{2}(z_i - v_i)^2}\right)$$

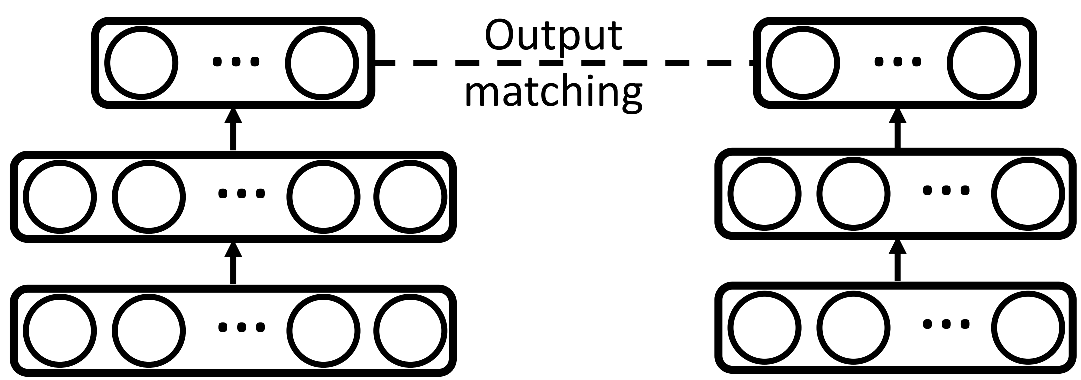
scaling

- **At high temperatures, the objective is equivalent to a quadratic function.**
  - **Distillation pays much more attention to logits that are negative than the average.**
  - This is potentially advantageous because these logits (which are not the correct label) are almost completely unconstrained by the classification loss.

- MNIST experiments
  - Hand-written digits (28x28 grayscale images)
  - 60000 training, 10000 test images
  - Source model: 2 hidden layers MLP with 1200 hidden nodes
  - Target model: 2 hidden layers MLP with 800 hidden nodes



Model	Error rate (%)
Source model	0.67
Target model (without knowledge distillation)	1.46
Target model (with knowledge distillation, $T = 20$)	0.74

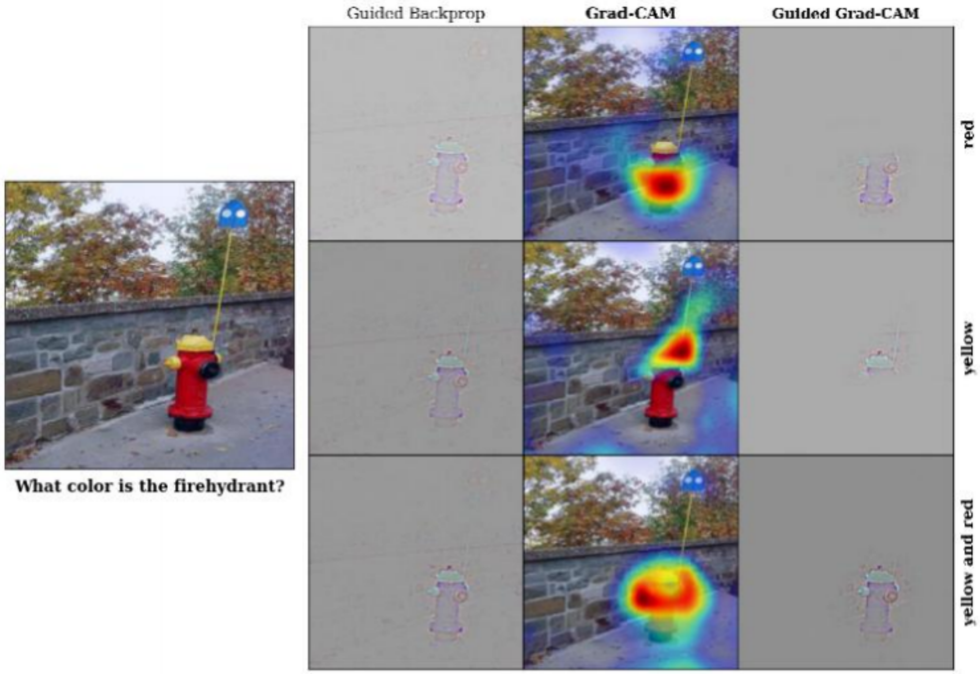


- Smaller target models get advantages by following larger source models

- Useful when target and source datasets/tasks are same
  - Performance may degrade when apply target dataset or task are changed

- **Main challenges:** what, when, and where to transfer
  - Decide the **form** of transferring knowledge
  - Decide **when** does transfer helps
  - Decide **which level** representations (layers) to transfer

- Visualizing **attention maps** in deep CNN is an open problem.
- Recently, a number of methods was proposed to improve attention maps.
  - e.g. Guided backpropagation [Springenberg et al., 2015], Grad-CAM[Selvaraju et al., 2016].
- In CNN models, the attention maps produced by intermediate features can be transferable knowledge.

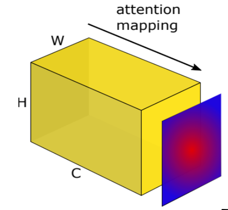


Visualization of VQA model.

* source: [Selvaraju et al., 2016]

- Matching the attention of intermediate features [Zagoruyko et al. 2017]
  - Make a 2D attention map from feature activations with attention mapping function F

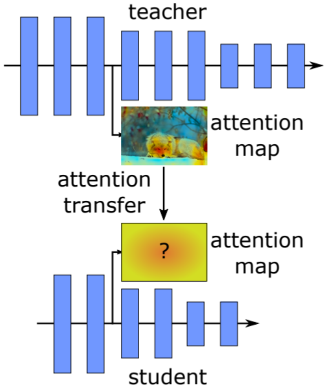
$$F(A_{h,w}) = \sum_{c=1}^{C} |A_{c,h,w}|^p$$

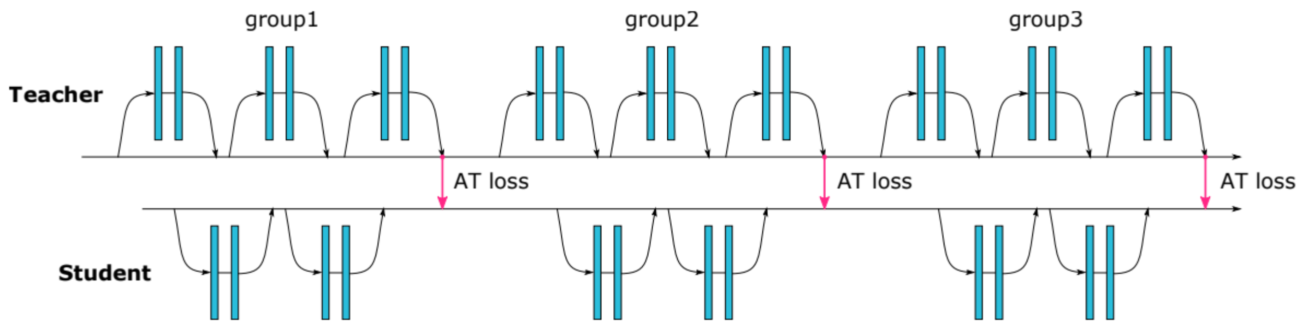


  - $p > 1$, feature activation $A_{c,h,w} \in \mathbb{R}^{C\times H\times W}$ (C channels, spatial size $H \times W$)
  - Train the original loss with the attention map matching regularization term

$$\mathcal{L}_{\text{at}}(\theta|\mathcal{D}) = \mathcal{L}_{\text{org}}(\theta|\mathcal{D}) + \frac{\beta}{2}\sum_{j\in\mathcal{I}}\left\| \frac{Q_{\mathcal{T}}^j(\theta,x)}{\left\|Q_{\mathcal{T}}^j(\theta,x)\right\|_2} - \frac{Q_{\mathcal{S}}^j(\theta,x)}{\left\|Q_{\mathcal{S}}^j(\theta,x)\right\|_2}\right\|_p$$

where $Q_T^j = vec(F(A_T^j))$ and $Q_S^j = vec(F(A_S^j))$ are respectively the j-th pair of target (student) and source (teacher) attention maps.





* source: [Zagoryuko et al., 2017]

Attention Transfer: Experimental Results

- Attention transfer works better than original distillation methods or they can be used together
  - Hyper-parametric choices:
    - Choose proper attention mapping function
    - Layers to transfer the attention map

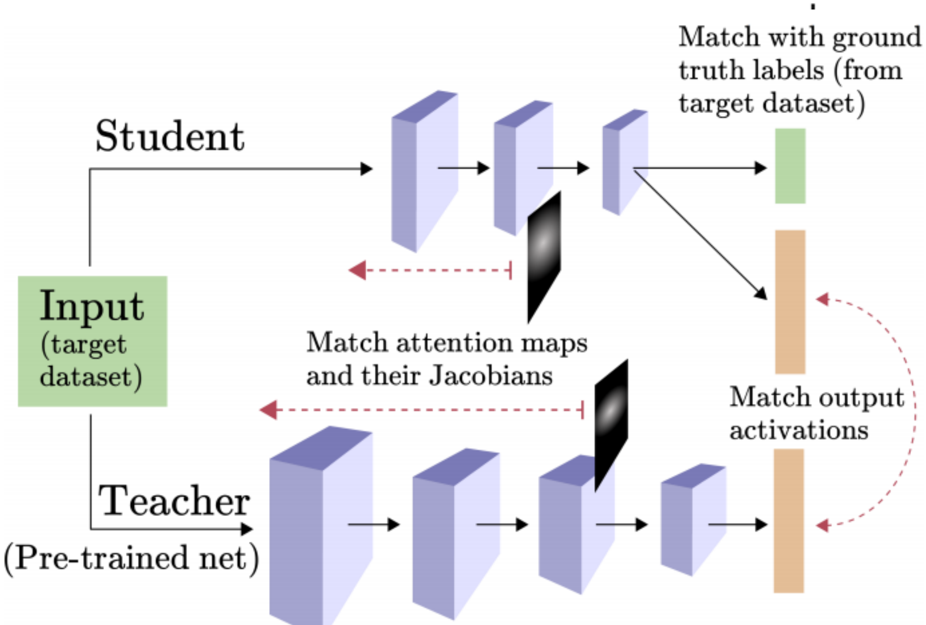
student	teacher	student	AT	F-ActT	KD	AT+KD	teacher
NIN-thin, 0.2M	NIN-wide, 1M	9.38	8.93	9.05	8.55	8.33	7.28
WRN-16-1, 0.2M	WRN-16-2, 0.7M	8.77	7.93	8.51	7.41	7.51	6.31
WRN-16-1, 0.2M	WRN-40-1, 0.6M	8.77	8.25	8.62	8.39	8.01	6.58
WRN-16-2, 0.7M	WRN-40-2, 2.2M	6.31	5.85	6.24	6.08	5.71	5.23

CIFAR-10 experiments. **AT**: attention transfer, **F-ActT**: full activation transfer, **KD**: knowledge distillation **AT+KD**: applying AT and KD at the same time. AT+KD is best in most cases (for student networks)

type	model	ImageNet→CUB	ImageNet→Scenes
student	ResNet-18	28.5	28.2
KD	ResNet-18	27 (-1.5)	28.1 (-0.1)
AT	ResNet-18	27 (-1.5)	27.1 (-1.1)
teacher	ResNet-34	26.5	26

Large-scale experiments. Using ImageNet pre-trained model, fine-tune source model with target dataset. Then, transfer to student model learning same target task.

- Several Jacobian-based regularizations have been proposed recently
  - Sobolev training [Czarnecki et al., 2017] demonstrated that using higher order (typically 1st order) derivatives along with the targets can help training.
  - [Srinivas et al., 2018] showed that matching Jacobians is a special case of previous distillation methods, when noise is added to the inputs.
- They added a new branch for distillation, and matched the output activations, attention maps, and their Jacobians (for the largest value of an attention map).



$$\mathcal{L}_{\text{kd}} = (1-\alpha)\mathcal{L}_{\text{ce}}(\mathbf{y},\mathbf{q}) + \alpha\mathcal{L}_{\text{ce}}(\mathbf{p}_T,\mathbf{q}_T)$$

$$\mathcal{L}_{\text{at}} = \sum_{j\in\mathcal{I}} \left\| \frac{Q^j_{\mathcal{T}}(\theta,x)}{\|Q^j_{\mathcal{T}}(\theta,x)\|_2} - \frac{Q^j_{\mathcal{S}}(\theta,x)}{\|Q^j_{\mathcal{S}}(\theta,x)\|_2} \right\|_p$$

$$\mathcal{L}_{\text{Jac}} = \sum_{j\in\mathcal{I}} \left\| \frac{\nabla_x f^j(x)}{\|\nabla_x f^j(x)\|_2} - \frac{\nabla_x g^j(x)}{\|\nabla_x g^j(x)\|_2} \right\|_p$$

where f^j, g^j are max points of j-th attention maps of target and source model, respectively.

...wise linear function;

The correct hypothesis is enriched with derivative information.

* source: [Czarnecki et al., 2017], [Srinivas et al., 2018]

- Matching Jacobians improves distillation performance in small data.

Distillation performance on the CIFAR100 dataset

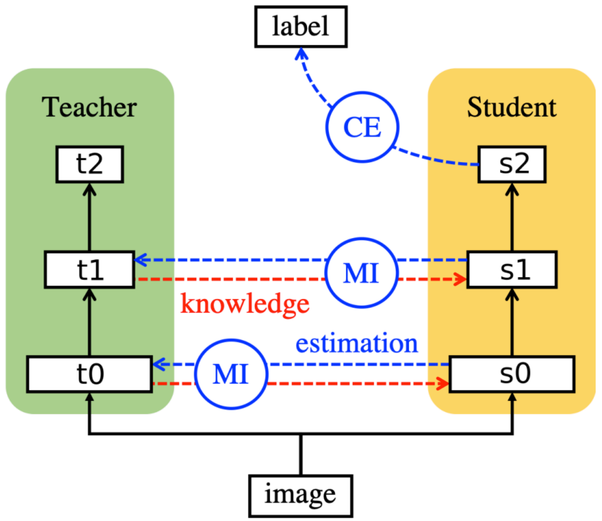
# of Data points per class →	1	5	10	50	100	500 (full)
Cross-Entropy (CE) training	5.69	13.9	20.03	37.6	44.92	54.28
CE + match activations	12.13	26.97	33.92	46.47	50.92	**56.65**
CE + match Jacobians	6.78	23.94	32.03	45.71	51.47	53.44
CE + match {activations + Jacobians}	**13.78**	**33.39**	**39.55**	**49.49**	**52.43**	54.57
Match activations only	10.73	28.56	33.6	45.73	50.15	56.59
Match {activations + Jacobians}	13.09	33.31	38.16	47.79	50.06	51.33

- Matching Jacobians improves performance of all case of transfer learning.
- None of the methods match the oracle performance of pre-trained model.

Transfer performance from Imagenet to MIT Scenes dataset

# of Data points per class →	5	10	25	50	Full
Cross-Entropy (CE) training on untrained student network	11.64	20.30	35.19	46.38	59.33
CE on pre-trained student network (Oracle)	**25.93**	**43.81**	**57.65**	**64.18**	**71.42**
CE + match activations (Li & Hoiem, 2016)	17.08	27.13	45.08	55.22	65.22
CE + match {activations + Jacobians}	17.88	28.25	45.26	56.49	66.04
CE + match {activations + attention} (Zagoruyko & Komodakis, 2017)	16.53	28.35	46.01	57.80	**67.24**
CE + match {activations + attention + Jacobians}	**18.02**	**29.25**	**47.31**	**58.35**	**67.31**

- [Ahn et al., 2019] **maximize mutual information** between source/target models
  - Use the **variational information maximization** [Barber and Agakov, 2003]
  - Instead of matching a specific form of feature representations



variational information maximization

$$
\begin{aligned}
I(t;s) &= H(t) - H(t|s) \\
&= H(t) + \mathbb{E}_{t,s}[\log p(t|s)] \\
&= H(t) + \mathbf{E}_{t,s}[\log q(t|s)] + \mathbf{E}_{s}[D_{\mathrm{KL}}(p(t|s)||q(t|s))] \\
&\geq H(t) + \mathbf{E}_{t,s}[\log q(t|s)]
\end{aligned}
$$

- Use a Gaussian distribution for modeling $q(t|s)$ with heteroscedastic mean $\mu(s)$ and homoscedastic variance $\sigma(s)$

$$
-\log q(t|s) = \sum_{c,h,w} \log \sigma_c + \frac{(t_{c,h,w} - \mu_{c,h,w}(s))^2}{2\sigma_c^2} + \text{constant}
$$

* source: [Ahn et al., 2019]

- Apply **Variational Information Distillation** (VID) to different locations
  - **VID-I**: between intermediate layers of teacher/student networks
  - **VID-LP**: between penultimate layers of teacher/student networks

Knowledge Distillation on CIFAR-10

M	5000	1000	500	100
Teacher	94.26	-	-	-
Student	90.72	84.67	79.63	58.84
KD	91.27	86.11	82.23	64.24
FitNet	90.64	84.78	80.73	68.90
AT	91.60	87.26	84.94	73.40
NST	91.16	86.55	82.61	64.53
VID-I	**91.85**	**89.73**	**88.09**	**81.59**
KD + AT	91.81	87.34	85.01	76.29
KD + VID-I	91.7	88.59	86.53	78.48

Transfer learning from ImageNet to CUB200

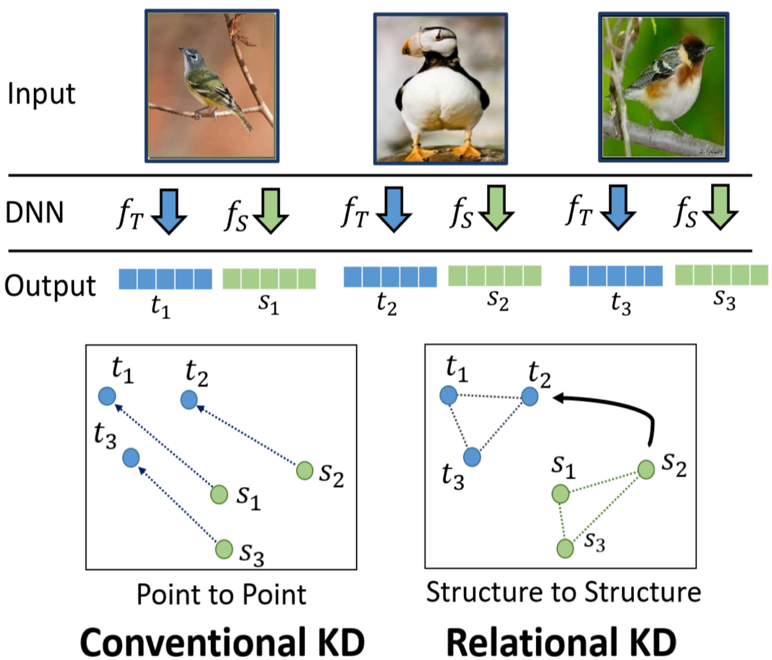
M	≈29.95	20	10	5
Student	37.22	24.33	12.00	7.09
fine-tuning	76.69	71.00	59.25	44.07
LwF	55.18	42.13	26.23	14.27
FitNet	66.63	56.63	46.68	31.04
AT	54.62	41.44	28.90	16.55
NST	55.01	41.87	23.76	15.63
VID-LP	65.59	54.12	39.20	27.86
VID-I	**73.25**	**67.20**	**56.86**	**46.21**
LwF + FitNet	68.69	58.81	48.86	31.30
VID-LP + VID-I	69.71	63.94	52.87	41.12

- **VID** can be applied between CNNs/MLPs
  - VID achieves state-of-the-art performance compared to other MLPs on CIFAR-10

Network	MLP-4096	MLP-2048	MLP-1024
Student	70.60	70.78	70.90
KD	70.42	70.53	70.79
FitNet	76.02	74.08	72.91
VID-I	**85.18**	**83.47**	**78.57**
Urban *et al.* [27]		74.32	
Lin *et al.* [17]		78.62	

- [Park et al., 2019] transfers the mutual **relations of data examples**
    - Knowledge distillation (KD) only mimic the output of individual data point

- Author considers two types of relations: **distance & angle**



Distance: L2 distance

$$\psi_{\mathrm{D}}(t_i, t_j) = \frac{1}{\mu} \|t_i - t_j\|_2 ,$$

$$\mathcal{L}_{\text{RKD-D}} = \sum_{(x_i, x_j) \in \mathcal{X}^2} l_\delta\big(\psi_{\mathrm{D}}(t_i, t_j), \psi_{\mathrm{D}}(s_i, s_j)\big),$$

Angle: Cosine similarity

$$\psi_{\mathrm{A}}(t_i, t_j, t_k) = \cos \angle t_i t_j t_k = \langle \mathbf{e}^{ij}, \mathbf{e}^{kj} \rangle$$

$$\text{where} \quad \mathbf{e}^{ij} = \frac{t_i - t_j}{\|t_i - t_j\|_2}, \mathbf{e}^{kj} = \frac{t_k - t_j}{\|t_k - t_j\|_2}.$$

$$\mathcal{L}_{\text{RKD-A}} = \sum_{(x_i, x_j, x_k) \in \mathcal{X}^3} l_\delta\big(\psi_{\mathrm{A}}(t_i, t_j, t_k), \psi_{\mathrm{A}}(s_i, s_j, s_k)\big),$$

l_δ: feature matching loss (Huber, L2 etc.)

* source: [Park et al., 2019]

Relational Knowledge Distillation: Experimental Results

- Apply three types of **relational knowledge distillation (RKD)**
  - **RKD-D**: only considers distance relationship
  - **RKD-A**: only considers angular relationship
  - **RKD-DA**: considers both, distance and angular relationship

	Baseline (Triplet [31])	FitNet [27]	Attention [47]	DarkRank [7]	Ours RKD-D	Ours RKD-A	Ours RKD-DA
$\ell 2$ normalization	O	O	O	O	O / X	O / X	O / X
ResNet18-16	37.71	42.74	37.68	46.84	46.34 / 48.09	45.59 / **48.60**	45.76 / 48.14
ResNet18-32	44.62	48.60	45.37	53.53	52.68 / **55.72**	53.43 / 55.15	53.58 / 54.88
ResNet18-64	51.55	51.92	50.81	56.30	56.92 / 58.27	56.77 / 58.44	57.01 / **58.68**
ResNet18-128	53.92	54.52	55.03	57.17	58.31 / 60.31	58.41 / **60.92**	59.69 / 60.67
ResNet50-512	61.24						

Recall@1 on CUB-200 dataset. The teacher is ResNet50-512 (model-d refers dimension)

	CIFAR-100 [15]	Tiny ImageNet [46]
Baseline	71.26	54.45
RKD-D	72.27	54.97
RKD-DA	72.97	56.36
HKD [11]	74.26	57.65
HKD+RKD-DA	**74.66**	**58.15**
FitNet [27]	70.81	55.59
FitNet+RKD-DA	72.98	55.54
Attention [47]	72.68	55.51
Attention+RKD-DA	73.53	56.55
Teacher	77.76	61.55

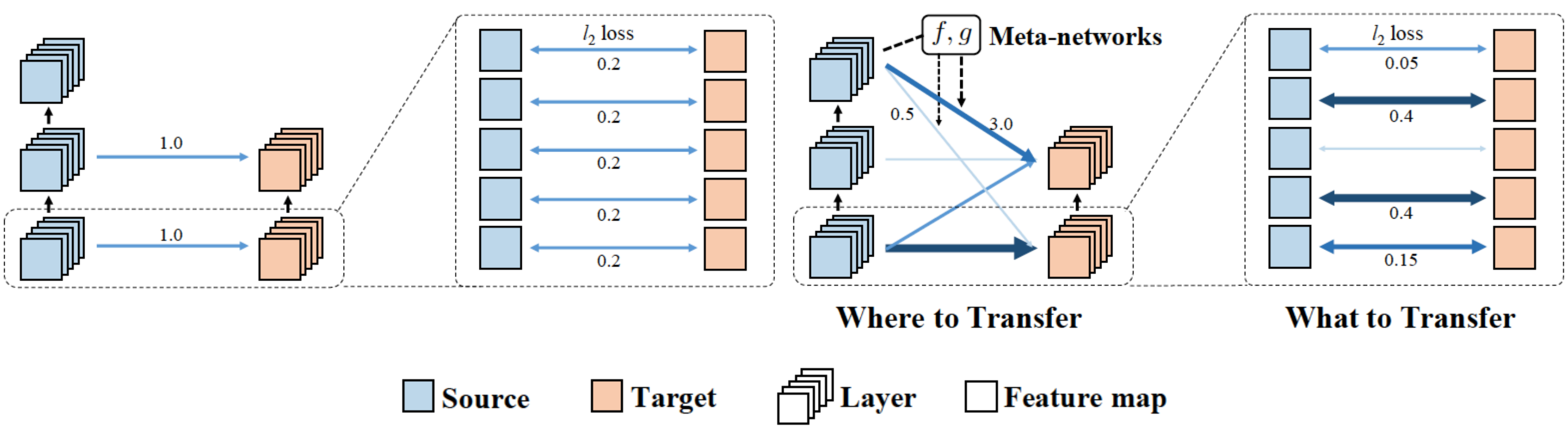
Accuracy (%) on CIFAR-100 and Tiny ImageNet.

Teacher: ResNet-50, student: VGG11
HKD: Conventional knowledge distillation

- Previous methods **transfer hand-crafted and fixed source knowledge**
  - **Hand-crafted matching formulations**
    - E.g., **KL divergence** [Hinton et al., 2015] between output layers, **attention map** [Zagoruyko et al. 2017] between hidden feature maps
  - **Hand-crafted matching connections**
    - Transfer on output activations of each group of residual/convolutional blocks

- [Jang et al., 2019] automatically **find what and where to transfer based on meta-learning** for maximizing transfer effect

Previous methods | **Learning What and Where to Transfer (L2T-ww)**



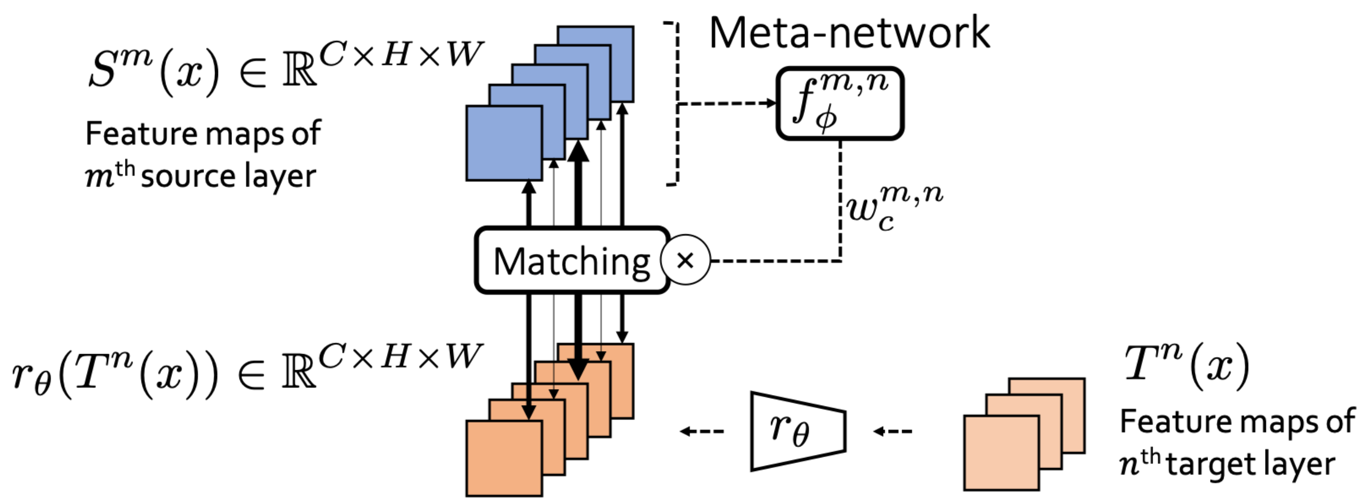
* source: [Jang et al., 2019]

- [Jang et al., 2019] use **meta-weighted feature matching** for transfer
- **Meta-network f** decides **useful channels** to transfer

Weight for channel c

$$\mathcal{L}_{\texttt{wfm}}^{m,n}(\theta|x,w^{m,n}) = \frac{1}{HW}\sum_{c} w_c^{m,n} \sum_{i,j} \left(r_\theta(T_\theta^n(x))_{c,i,j} - S^m(x)_{c,i,j}\right)^2$$

L2 distance at channel c



* source: [Jang et al., 2019]

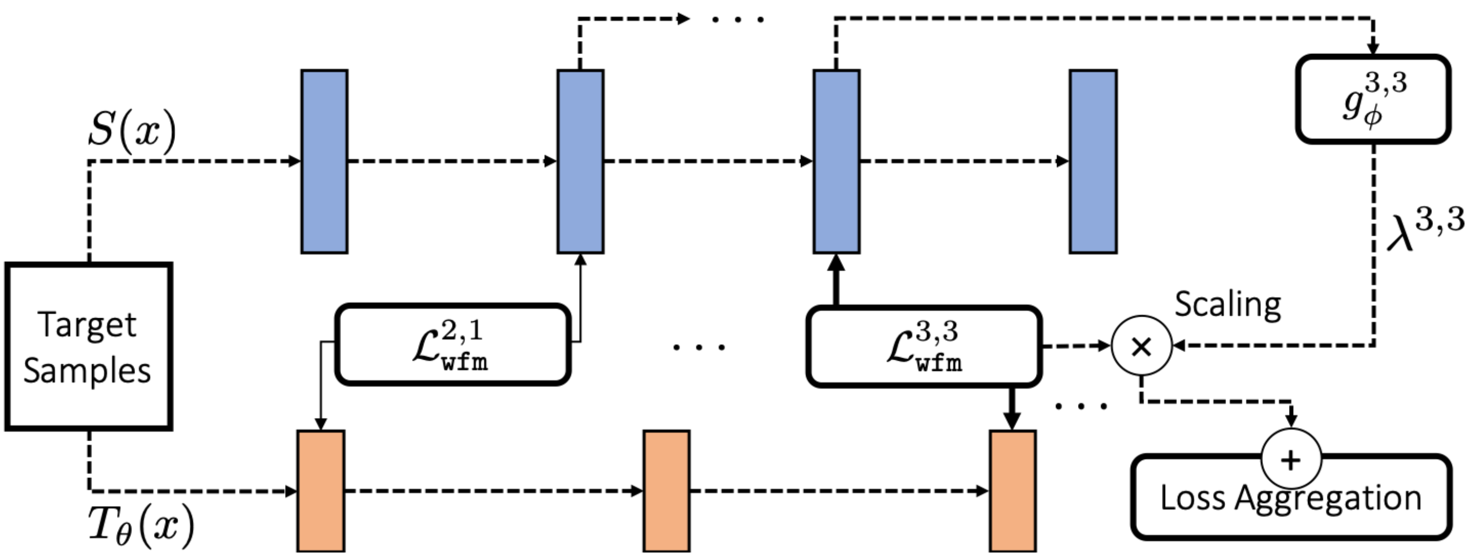
- [Jang et al., 2019] use **meta-weighted feature matching** for transfer

- **Meta-network g** decides **useful pairs** of source/target layers to transfer

Weight for pair (m, n)

$$\mathcal{L}_{\texttt{wfm}}(\theta|x,\phi) = \sum_{(m,n)\in\mathcal{C}} \lambda^{m,n} \mathcal{L}^{m,n}_{\texttt{wfm}}(\theta|x, w^{m,n})$$

Transfer loss on pair (m, n)



* source: [Jang et al., 2019]

Q) How to learn meta-networks f, g?

- [Jang et al., 2019] propose a bilevel scheme for training meta-parameters ϕ of meta-networks f, g

1. *Knowledge transfer*: for $t = 1, \dots, T$,
$$\theta_{t+1} = \theta_t - \alpha \nabla_\theta \mathcal{L}_{\texttt{wfm}}(\theta_t | x, \phi)$$ ← Transfer loss
2. *One-step adaption*:
$$\theta_{T+2} = \theta_{T+1} - \alpha \nabla_\theta \mathcal{L}_{\texttt{org}}(\theta_{T+1} | x, y)$$
3. *Evaluation*:
$$\mathcal{L}_{\texttt{meta}}(\phi) = \mathcal{L}_{\texttt{org}}(\theta_{T+2} | x, y).$$ ← Task-specific loss
4. Update ϕ based on $\nabla_\phi \mathcal{L}_{\texttt{meta}}(\phi)$ using second-order gradients

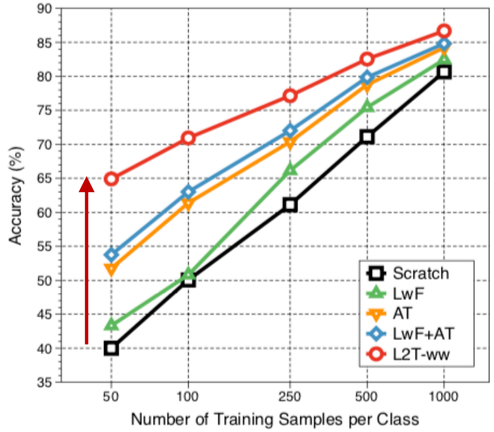
- Effective for learning ϕ with **a small number of steps T**
  - A popular bilevel scheme [Franceschi et al., 2018] **requires many steps**
- Joint-learning θ and ϕ without separate meta-learning phase

- **L2T-ww outperforms** previous methods **on various datasets, architectures**

Source task	TinyImageNet		ImageNet			
Target task	CIFAR-100	STL-10	CUB200	MIT67	Stanford40	Stanford Dogs
Scratch	67.69±0.22	65.18±0.91	42.15±0.75	48.91±0.53	36.93±0.68	58.08±0.26
LwF[6]	69.23±0.09	68.64±0.58	45.52±0.66	53.73±2.14	39.73±1.63	66.33±0.45
AT[1] (one-to-one)	67.54±0.40	74.19±0.22	57.74±1.17	59.18±1.57	59.29±0.91	69.70±0.08
LwF[6]+AT[1] (one-to-one)	68.75±0.09	75.06±0.57	58.90±1.32	61.42±1.68	60.20±1.34	72.67±0.26
FM[3] (single)	69.40±0.67	75.00±0.34	47.60±0.31	55.15±0.93	42.93±1.48	66.05±0.76
FM[3] (one-to-one)	69.97±0.24	76.38±1.18	48.93±0.40	54.88±1.24	44.50±0.96	67.25±0.88
L2T-w (single)	70.27±0.09	74.35±0.92	51.95±0.83	60.41±0.37	46.25±3.66	69.16±0.70
L2T-w (one-to-one)	70.02±0.19	76.42±0.52	56.61±0.20	59.78±1.90	48.19±1.42	69.84±1.45
L2T-ww (all-to-all)	**70.96**±0.61	**78.31**±0.21	**65.05**±1.19	**64.85**±2.75	**63.08**±0.88	**78.08**±0.96

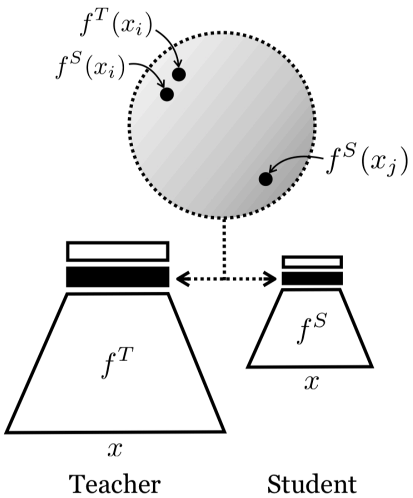
- **L2T-ww** can **aggregate multiple source** knowledge (left)
- **L2T-ww** can transfer knowledge effectively **on limited-data regime**

First source	TinyImageNet (ResNet32)			
Second source	None	TinyImageNet (ResNet20)	TinyImageNet (ResNet32)	CIFAR-10 (ResNet32)
Scratch	65.18±0.91	65.18±0.91	65.18±0.91	65.18±0.91
LwF[6]	68.64±0.58	68.56±2.24	68.05±2.12	69.51±0.63
AT[1]	74.19±0.22	73.24±0.12	73.78±1.16	73.99±0.51
LwF[6]+AT[1]	75.06±0.57	74.72±0.46	74.77±0.30	74.41±1.51
FM[3] (single)	75.00±0.34	75.83±0.56	75.99±0.11	74.60±0.73
FM[3] (one-to-one)	76.38±1.18	77.45±0.48	77.69±0.79	77.15±0.41
L2T-ww (all-to-all)	**78.31**±0.21	**79.35**±0.41	**79.80**±0.52	**80.52**±0.29



* source: [Jang et al., 2019]

- [Tian et al., 2020] transfers the **output similarity** of data points
  - Maximize the similarity of **same data point**, and minimize between **other points**



$f^T(x_i)$ and $f^S(x_i)$ is similar (**same sample**)
$f^T(x_i)$ and $f^S(x_j)$ is not similar (**other $N-1$ samples**)

- Contrastive-object **maximize the mutual information** between models

$$I(T;S) \geq \log(N) + \underbrace{\mathbb{E}_{q(T,S|C=1)}[\log h^*(T,S)]}_{\text{Maximize similarity}} + \underbrace{N\mathbb{E}_{q(T,S|C=0)}[\log(1-h^*(T,S))]}_{\text{Minimize similarity}}$$

$$h(T,S) = \frac{e^{g^T(T)'g^S(S)/\tau}}{e^{g^T(T)'g^S(S)/\tau} + \frac{N}{M}}$$

$h(T,S) \in [0,1]$ is a similarity measure
Where $T = f^T(x_i)$, $S = f^S(x_j)$ is the representation
and g^T, g^S is a linear layer of teacher and student, respectively

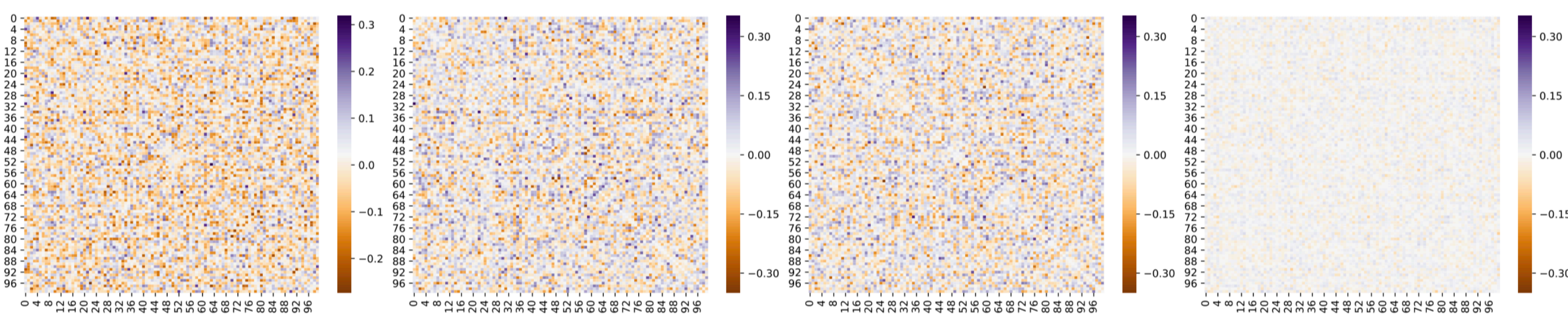
* source: [Tian et al., 2020]

Contrastive Representation Distillation: Experimental Results

- **CRD consistently outperforms** previous methods **on various architectures**

Teacher Student	WRN-40-2 WRN-16-2	WRN-40-2 WRN-40-1	resnet56 resnet20	resnet110 resnet20	resnet110 resnet32	resnet32x4 resnet8x4	vgg13 vgg8
Teacher	75.61	75.61	72.34	74.31	74.31	79.42	74.64
Student	73.26	71.98	69.06	69.06	71.14	72.50	70.36
KD*	74.92	73.54	70.66	70.67	73.08	73.33	72.98
FitNet*	73.58 (↓)	72.24 (↓)	69.21 (↓)	68.99 (↓)	71.06 (↓)	73.50 (↑)	71.02 (↓)
AT	74.08 (↓)	72.77 (↓)	70.55 (↓)	70.22 (↓)	72.31 (↓)	73.44 (↑)	71.43 (↓)
SP	73.83 (↓)	72.43 (↓)	69.67 (↓)	70.04 (↓)	72.69 (↓)	72.94 (↓)	72.68 (↓)
CC	73.56 (↓)	72.21 (↓)	69.63 (↓)	69.48 (↓)	71.48 (↓)	72.97 (↓)	70.71 (↓)
VID	74.11 (↓)	73.30 (↓)	70.38 (↓)	70.16 (↓)	72.61 (↓)	73.09 (↓)	71.23 (↓)
RKD	73.35 (↓)	72.22 (↓)	69.61 (↓)	69.25 (↓)	71.82 (↓)	71.90 (↓)	71.48 (↓)
PKT	74.54 (↓)	73.45 (↓)	70.34 (↓)	70.25 (↓)	72.61 (↓)	73.64 (↑)	72.88 (↓)
AB	72.50 (↓)	72.38 (↓)	69.47 (↓)	69.53 (↓)	70.98 (↓)	73.17 (↓)	70.94 (↓)
FT*	73.25 (↓)	71.59 (↓)	69.84 (↓)	70.22 (↓)	72.37 (↓)	72.86 (↓)	70.58 (↓)
FSP*	72.91 (↓)	n/a	69.95 (↓)	70.11 (↓)	71.89 (↓)	72.62 (↓)	70.23 (↓)
NST*	73.68 (↓)	72.24 (↓)	69.60 (↓)	69.53 (↓)	71.96 (↓)	73.30 (↓)	71.53 (↓)
CRD	**75.48** (↑)	**74.14** (↑)	**71.16** (↑)	**71.46** (↑)	**73.48** (↑)	**75.51** (↑)	**73.94** (↑)
CRD+KD	75.64 (↑)	74.38 (↑)	71.63 (↑)	71.56 (↑)	73.75 (↑)	75.46 (↑)	74.29 (↑)

- Visualization: ***difference*** of correlation matrices of student and teacher logits.
  - CRD shows **significant matching** between **student's and teacher's correlations**

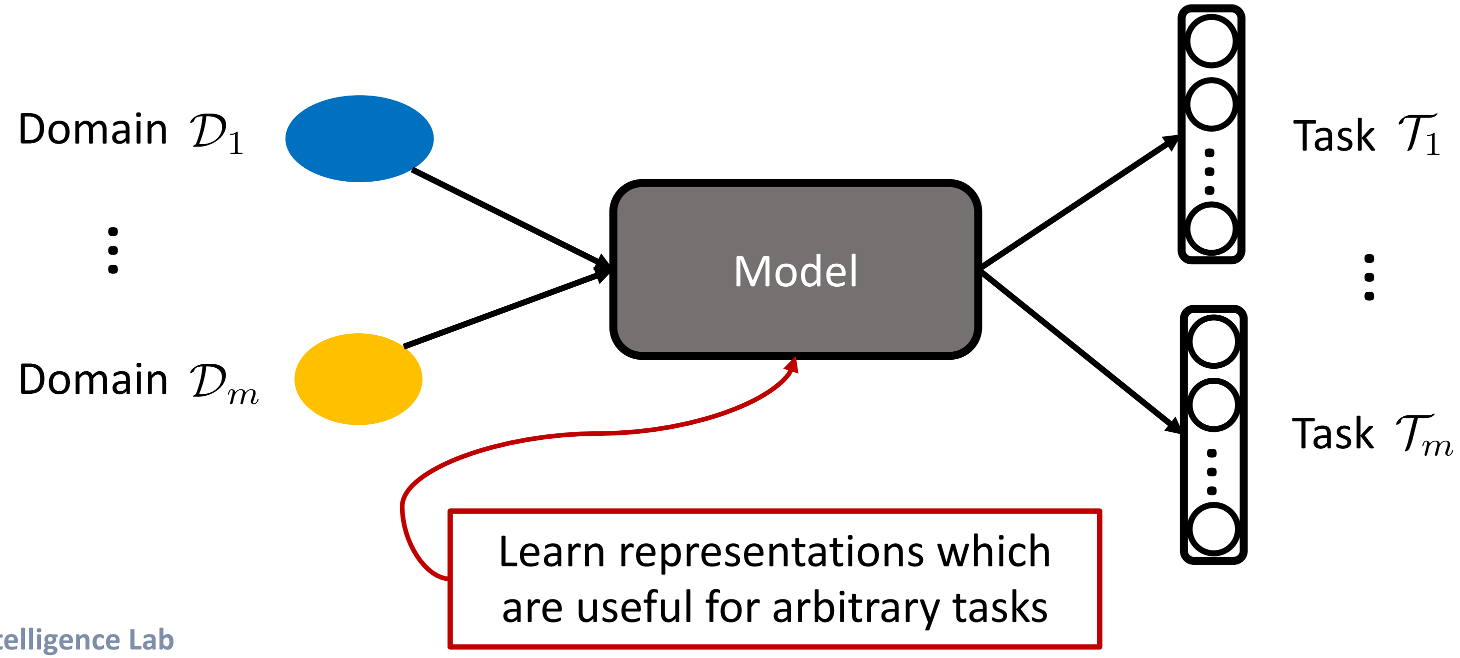


(a) Student: vanilla (b) Student: AT (c) Student: KD (d) Student: ours (CRD)

Table of Contents

- Definition of multi-task learning [Zhang and Yang, 2017]
  - Given ***m* learning *tasks*** $\{\mathcal{T}_i\}_{i=1}^m$
    - where all the tasks or a subset of them are **related**,
  - ***Multi-task learning*** (MTL) aims to **improve** the learning of a model for $\mathcal{T}_i$ using the knowledge contained in all or some of the m tasks

  - In the view of definition of transfer learning [Pan et al., 2010], all learning tasks $\{\mathcal{T}_i\}_{i=1}^m$ are considered as both source and target tasks

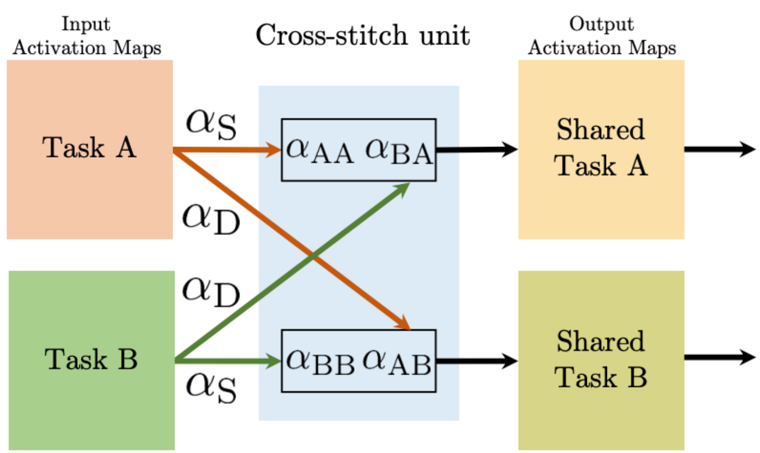


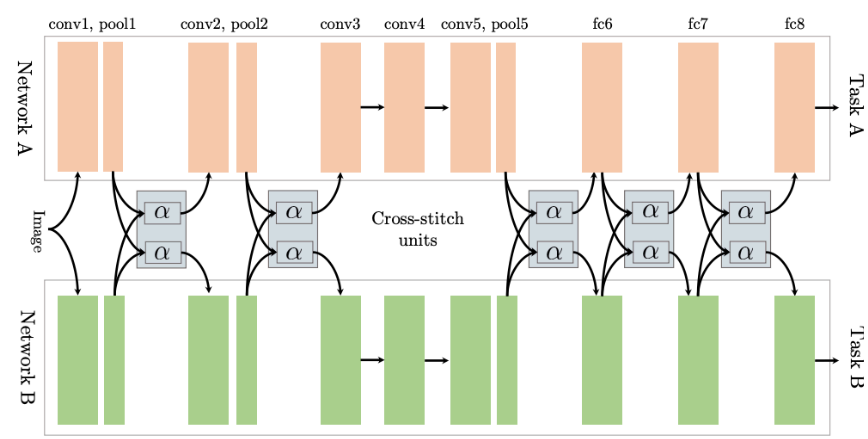
Cross-stitch Networks for Multi-task Learning

- Cross-stitch units [Misra et al., 2016] try to find the best shared representations for multi-task learning

$$\begin{bmatrix} \tilde{x}_{\mathrm{A}}^{ij} \\ \tilde{x}_{\mathrm{B}}^{ij} \end{bmatrix} = \begin{bmatrix} \alpha_{\mathrm{AA}} & \alpha_{\mathrm{AB}} \\ \alpha_{BA} & \alpha_{\mathrm{BB}} \end{bmatrix} \begin{bmatrix} x_{\mathrm{A}}^{ij} \\ x_{\mathrm{B}}^{ij} \end{bmatrix}$$

  - $x_{\mathrm{A}}^{ij}, \ x_{\mathrm{B}}^{ij}$ are activation map (at location i,j) of networks for task A, B, respectively
  - α is trained by backpropagation with different learning rates
  - Maintain one cross-stitch unit per channel





* source: [Misra et al., 2016]

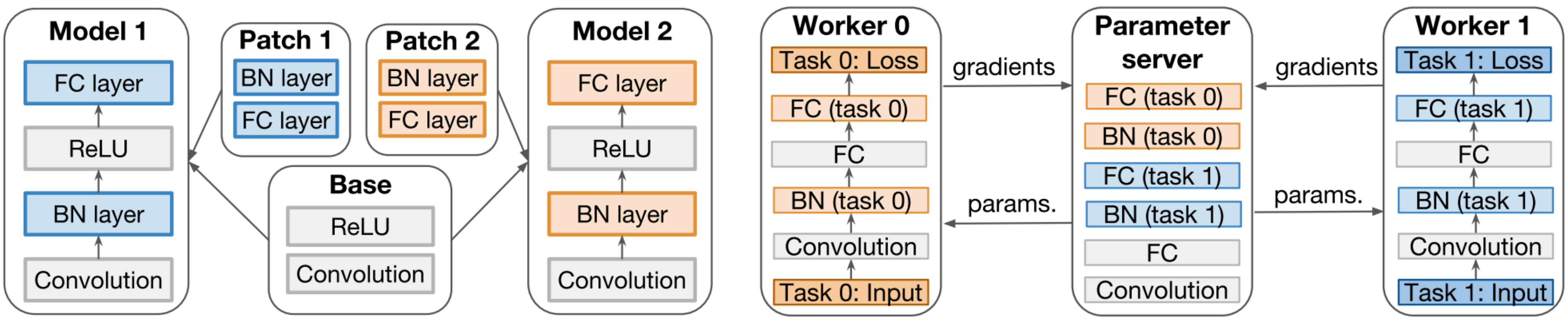
Cross-stitch Networks for Multi-task Learning

- Multi-task (Surface Normal / Segmentation) learning on NYU-v2 dataset
  - Cross-stitch uses 2 convolutional networks
  - Ensemble uses 4 convolutional networks (2 for each task)
  - It shows that **sharing information can improve** the performance

	Surface Normal					Segmentation		
	Angle Distance (Lower Better)		Within $t°$ (Higher Better)			(Higher Better)		
Method	Mean	Med.	11.25	22.5	30	pixacc	mIU	fwIU
One-task	34.8	19.0	38.3	53.5	59.2	-	-	-
	-	-	-	-	-	46.6	18.4	33.1
Ensemble	34.4	18.5	38.7	54.2	59.7	-	-	-
	-	-	-	-	-	**48.2**	18.9	33.8
Split `conv4`	34.7	19.1	38.2	53.4	59.2	47.8	19.2	33.8
MTL-shared	34.7	18.9	37.7	53.5	58.8	45.9	16.6	30.1
Cross-stitch [**ours**]	**34.1**	**18.2**	**39.0**	**54.4**	**60.2**	47.2	**19.3**	**34.0**

- **Drawbacks**
  - Parameter-inefficiency because it requires **one CNN per each task**

- One **model-patch** [Mudrakarta et al., 2019] for each task
  - One shared base model for all tasks
  - For multi-task learning, train model-patches and shared parts simultaneously
  - For transfer learning, freeze the shared parts / train new model-patch only
  - **Multiple networks share most weights** (>95% parameters)



- Two types of **model-patch**
  - **Scale-and-bias** (S/B) patch: a normalization layer (e.g., BN)
  - **Depth-wise-convolution** (DW) patch: depth-wise separable convolutional layers

* source: [Mudrakarta et al., 2019]

- **Despite using much fewer parameters**, competitive performance is achieved

Table 4: Multi-task learning with MobilenetV2 on ImageNet and Places-365.

Task	S/B patch + last layer	Last layer	Independently trained
Imagenet Places365	70.2% **54.3%**	64.4% 51.4%	71.8% 54.2%
# total parameters	3.97M	3.93M	6.05M

One patch for each task
Sharing Most weights

≈

One model for each task

- When transfer learning, **despite fine-tuning much fewer parameters**, it achieves nontrivial performance

Fine-tuned params.	Flowers		Cars		Aircraft	
	Acc.	#params	Acc.	#params	Acc.	#params
Last layer	84.5	208K	55	402K	45.9	205K
S/B + last layer	**90.4**	244K	**81**	437K	**70.7**	241K
S/B only (random last)	79.5	**36K**	33	**36K**	52.3	**36K**
All (ours)	93.3	25M	92.3	25M	87.3	25M
All (Cui et al., 2018)	96.3	25M	91.3	25M	82.6	25M

Table of Contents

- The naive approach to combining multi objective losses is to perform a weighted linear sum of the losses for each individual task.

$$\mathcal{L}_{\text{total}} = \sum_i \boxed{w_i} \mathcal{L}_i$$

- [Kendall et al., 2018] proposed that homoscedastic (i.e. task-dependent) *uncertainty* can be used as a weight for losses in a multi-task learning problem.
  - They adapted a likelihood as below, with a **noise scalar** σ . Note that the probability distribution becomes uniform as $\sigma \to \infty$.

For classification tasks $\quad p(\mathbf{y}|\mathbf{f}^{\mathbf{W}}(\mathbf{x})) = \text{Softmax}(\frac{1}{\sigma^2}\mathbf{f}^{\mathbf{W}}(x))$

For regression tasks $\quad p(\mathbf{y}|\mathbf{f}^{\mathbf{W}}(\mathbf{x})) = \mathcal{N}(\mathbf{f}^{\mathbf{W}}(\mathbf{x}), \sigma^2)$

  - Let's assume that the total likelihood can be factorized over the each output, given some sufficient statistics.

$$p(\mathbf{y}_1, ..., \mathbf{y}_K|\mathbf{f}^{\mathbf{W}}(\mathbf{x})) = p(\mathbf{y}_1|\mathbf{f}^{\mathbf{W}}(\mathbf{x})) \ldots p(\mathbf{y}_K|\mathbf{f}^{\mathbf{W}}(\mathbf{x}))$$

- The log likelihood for output can be written as

For classification tasks
$$\log p(\mathbf{y}=c|\mathbf{f}^{\mathbf{W}}(\mathbf{x})) = \frac{1}{\sigma^2}\mathbf{f}_c^{\mathbf{W}}(\mathbf{x}) - \log \sum_{c'} \exp\left(\frac{1}{\sigma^2}\mathbf{f}_{c'}^{\mathbf{W}}(\mathbf{x})\right)$$
$$\mathcal{L}_{\text{cls}}(\mathbf{W}) = -\log \text{Softmax}(\mathbf{y}, \mathbf{f}^{\mathbf{W}}(\mathbf{x}))$$

For regression tasks
$$\log p(\mathbf{y}|\mathbf{f}^{\mathbf{W}}(\mathbf{x})) \propto -\frac{1}{2\sigma^2}||\mathbf{y}-\mathbf{f}^{\mathbf{W}}(\mathbf{x})||^2 - \log\sigma$$
$$\mathcal{L}_{\text{reg}}(\mathbf{W}) = ||\mathbf{y}-\mathbf{f}^{\mathbf{W}}(\mathbf{x})||^2$$

This constructions can be trivially extended to multiple outputs.

- If there are two regression tasks,

$$\mathcal{L}(\mathbf{W},\sigma_1,\sigma_2) = -\log p(\mathbf{y}_1,\mathbf{y}_2|\mathbf{f}^{\mathbf{W}}(\mathbf{x}))$$
$$\propto \frac{1}{2\sigma_1^2}||\mathbf{y}_1-\mathbf{f}^{\mathbf{W}}(\mathbf{x})||^2 + \frac{1}{2\sigma_2^2}||\mathbf{y}_2-\mathbf{f}^{\mathbf{W}}(\mathbf{x})||^2 + \log\sigma_1\sigma_2$$

weighted sum
$$= \boxed{\frac{1}{2\sigma_1^2}\mathcal{L}_{1,\text{reg}}(\mathbf{W}) + \frac{1}{2\sigma_2^2}\mathcal{L}_{2,\text{reg}}(\mathbf{W})} + \log\sigma_1\sigma_2$$

- If the 1st task is a regression task, and the 2nd one is a classification task,

$$\mathcal{L}(\mathbf{W},\sigma_1,\sigma_2) = -\log p(\mathbf{y}_1,\mathbf{y}_2=c|\mathbf{f}^{\mathbf{W}}(\mathbf{x}))$$
$$\propto \frac{1}{2\sigma_1^2}||\mathbf{y}_1-\mathbf{f}^{\mathbf{W}}(\mathbf{x})||^2 + \log\sigma_1 - \log p(\mathbf{y}_2=c|\mathbf{f}^{\mathbf{W}}(\mathbf{x}))$$
$$= \frac{1}{2\sigma_1^2}||\mathbf{y}_1-\mathbf{f}^{\mathbf{W}}(\mathbf{x})||^2 - \frac{1}{\sigma_2^2}\log\text{Softmax}(\mathbf{y}_2,\mathbf{f}^{\mathbf{W}}(\mathbf{x})) + \log\sigma_1 + \log\frac{\sum_{c'}\exp\left(\frac{1}{\sigma_2^2}\mathbf{f}_{c'}^{\mathbf{W}}(\mathbf{x})\right)}{\left(\sum_{c'}\exp\left(\mathbf{f}_{c'}^{\mathbf{W}}(\mathbf{x})\right)\right)^{\frac{1}{\sigma_2^2}}}$$

weighted sum
$$\approx \boxed{\frac{1}{2\sigma_1^2}\mathcal{L}_{1,\text{reg}}(\mathbf{W}) + \frac{1}{\sigma_2^2}\mathcal{L}_{2,\text{cls}}(\mathbf{W})} + \log\sigma_1 + \log\sigma_2 \quad \text{as} \quad \sigma_2 \to 1 \, .$$

- In practice, the log variance $s := \log \sigma^2$ is trained by the network .
  - This term is added to weighted sum of original multi-task losses.
- In experiments, there are three tasks:
  - Semantic segmentation (classification)
  - Instance segmentation (regression)
  - Depth regression (regression)

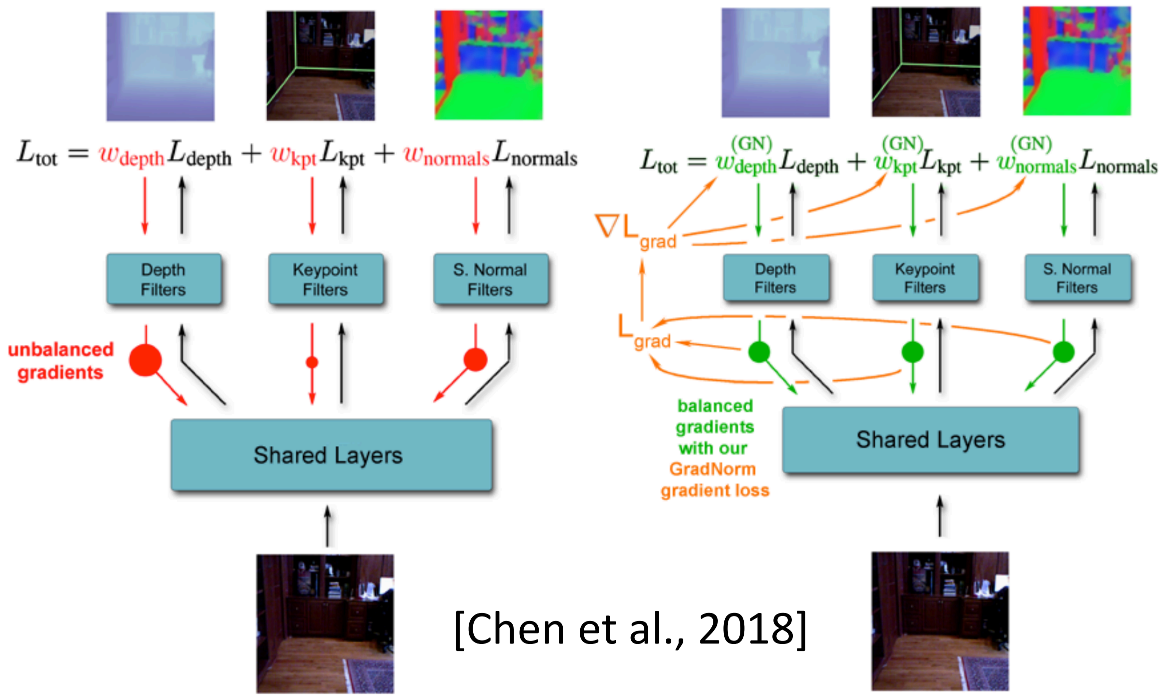
Approx. optimal weights are found by grid search.

Loss	Task Weights Seg.	Task Weights Inst.	Task Weights Depth	Segmentation IoU [%]	Instance Mean Error [px]	Inverse Depth Mean Error [px]
Segmentation only	1	0	0	59.4%	-	-
Instance only	0	1	0	-	4.61	-
Depth only	0	0	1	-	-	0.640
Unweighted sum of losses	0.333	0.333	0.333	50.1%	3.79	0.592
Approx. optimal weights	0.89	0.01	0.1	62.8%	3.61	0.549
2 task uncertainty weighting	✓	✓		61.0%	**3.42**	-
2 task uncertainty weighting	✓		✓	62.7%	-	0.533
2 task uncertainty weighting		✓	✓	-	3.54	0.539
3 task uncertainty weighting	✓	✓	✓	**63.4%**	3.50	**0.522**

* source: [Kendall et al., 2018]

Gradient Normalization for Adaptive Loss Balancing in Deep Multitask Networks

- At time t, the weighted average for multi-task learning = $\sum_i w_i(t)\mathcal{L}_i(t)$
- **The gradient for a task might be dominant** when multi-task learning
  - It depends on task difficulties, loss functions, and so on
  - **Q)** What is correct balance for w_i?



[Chen et al., 2018]

- **Key Idea**: If a task is **not trained enough** ⇒ norm of its gradient **should be large**

* source: [Chen et al., 2018]

- **Gradient norm**
  - $G_W^{(i)}(t) = \|\nabla_W w_i(t) L_i(t)\|_2$: gradient norm of task *i*
  - $\bar{G}_W(t) = \mathbb{E}_i[G_W^{(i)}(t)]$: average gradient norm across all tasks

- **Training rates** for measuring current states of learning of tasks
  - Inverse training rates $\tilde{L}_i(t) = L_i(t)/L_i(0)$
  - Relative inverse training rates $r_i(t) = \tilde{L}_i(t)/\mathbb{E}_j[\tilde{L}_j(t)]$

- Large $r_i(t) \Rightarrow$ need to train more ⇒ need large gradients
  - **Our desired gradient norm**:

$$G_W^{(i)}(t) \mapsto \bar{G}_W(t) \times [r_i(t)]^\alpha$$

    where α is a hyperparameter
  - To balance the norms based on training rates, minimize L_{grad} over w_i

$$L_{\text{grad}}(t; w_i(t)) = \sum_i \left| G_W^{(i)} - \bar{G}_W(t) \times [r_i(t)]^\alpha \right|$$

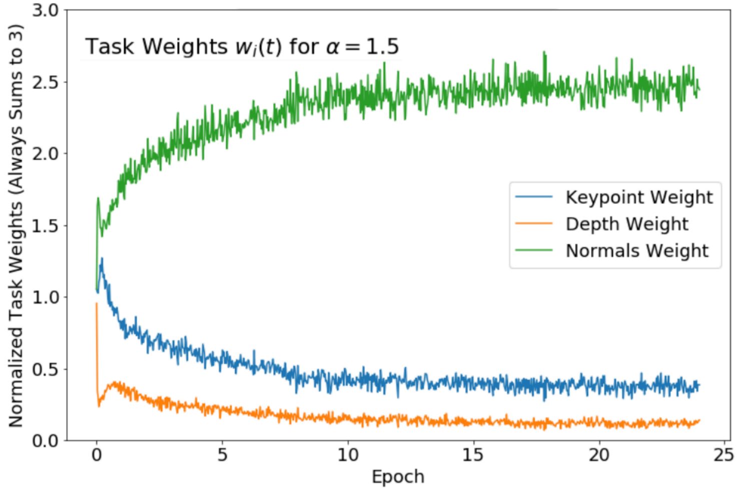
- Train on NYUv2+kepoint/segmentation dataset with 3 different tasks

Model and Weighting Method	Depth RMS Err. (m)	Seg. Err. (100-IoU)	Normals Err. (1-|cos|)
VGG Backbone			
Depth Only	1.038	-	-
Seg. Only	-	70.0	-
Normals Only	-	-	**0.169**
Equal Weights	0.944	70.1	0.192
GradNorm Static	0.939	**67.5**	0.171
GradNorm $\alpha = 1.5$	**0.925**	67.8	0.174

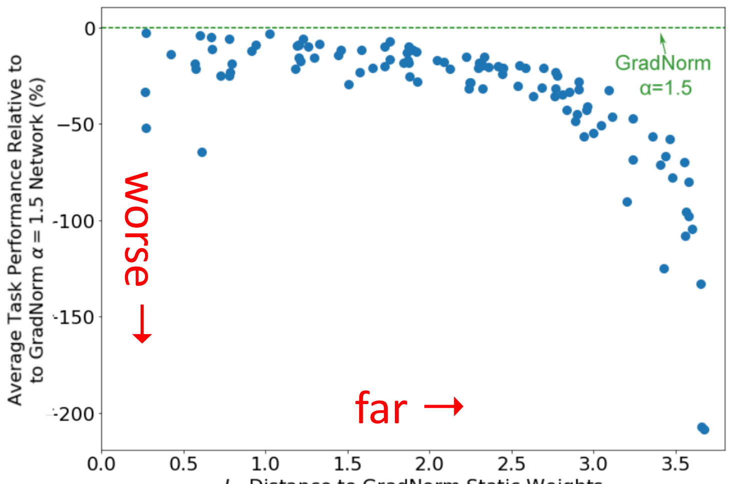
Model and Weighting Method	Depth RMS Err. (m)	Kpt. Err. (%)	Normals Err. (1-|cos|)
ResNet Backbone			
Depth Only	0.725	-	-
Kpt Only	-	7.90	-
Normals Only	-	-	**0.155**
Equal Weights	0.697	7.80	0.172
(Kendall et al., 2017)	0.702	7.96	0.182
GradNorm Static	0.695	7.63	0.156
GradNorm $\alpha = 1.5$	**0.663**	**7.32**	**0.155**

- If using farther weights from GradNorm, then worse results are obtained

Weights during training



Performance with various weights



Multi-task Learning as Multi-objective Optimization

- The loss function for multi-task learning is generally the weighted summation

$$\min_\theta \sum_{t=1}^{T} w_t \mathcal{L}_t(\theta)$$

- **For finding weights,** expensive grid search or heuristics are required
  - Heuristics: [Kendall et al., 2018], [Chen et al., 2018]

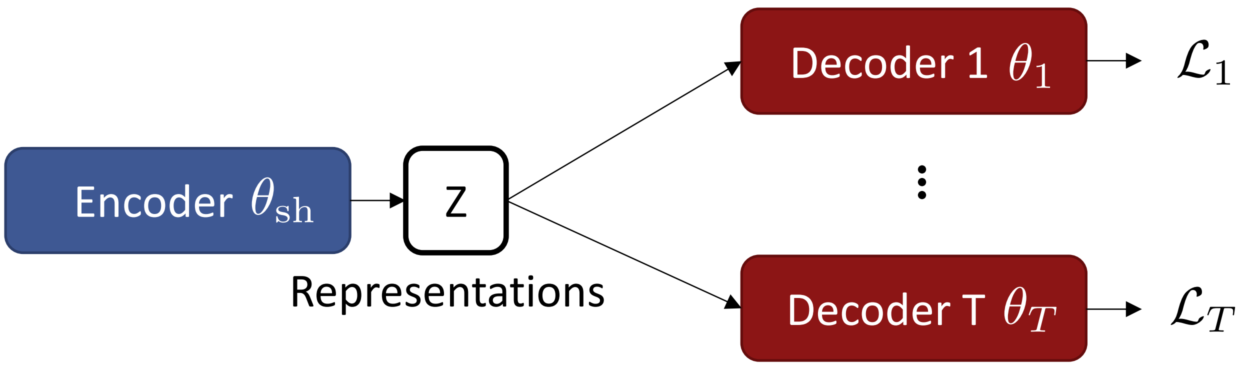
- **Pareto optimality** (multi-objective optimization formulation)
  - A solution θ *dominates* $\bar{\theta}$ if $\mathcal{L}_t(\theta) \leq \mathcal{L}_t(\bar{\theta})$ for all tasks t
  - A solution $\theta^\star$ is called *Pareto optimal* if there is no θ that dominates $\theta^\star$
  - The Pareto optimal solution can be considered as a solution for multi-task learning
  - **Q)** How to find the Pareto optimal solutions?

- **Multiple Gradient Descent Algorithm** (MGDA)

$$\min_{\alpha_1,\ldots,\alpha_T} \left\{ \left\| \sum_{t=1}^{T} \alpha_t \nabla_{\theta_{\text{sh}}} \mathcal{L}_t(\theta_{\text{sh}}, \theta_t) \right\|_2^2 \middle| \sum_{t=1}^{T} \alpha_t = 1, \alpha_t \geq 0 \right\}$$

  - Its solution gives **Pareto stationary** (necessary for optimality) **solutions or a descent direction that improves all tasks**
  - It can be efficiently solved by Frank-Wolfe algorithm (detail is omitted)

- **Issue:** MGDA needs to compute $\nabla_{\theta_{\mathrm{sh}}} \mathcal{L}_t(\theta_{\mathrm{sh}}, \theta_t)$ for each task t
    - Linear scaling of the training time
- **Solution:** Use encoder-decoder architectures
    - **One shared encoder** for all tasks
    - **One separate decoder** for each task
    - Encoder-decoder architectures are typically used for multi-task learning



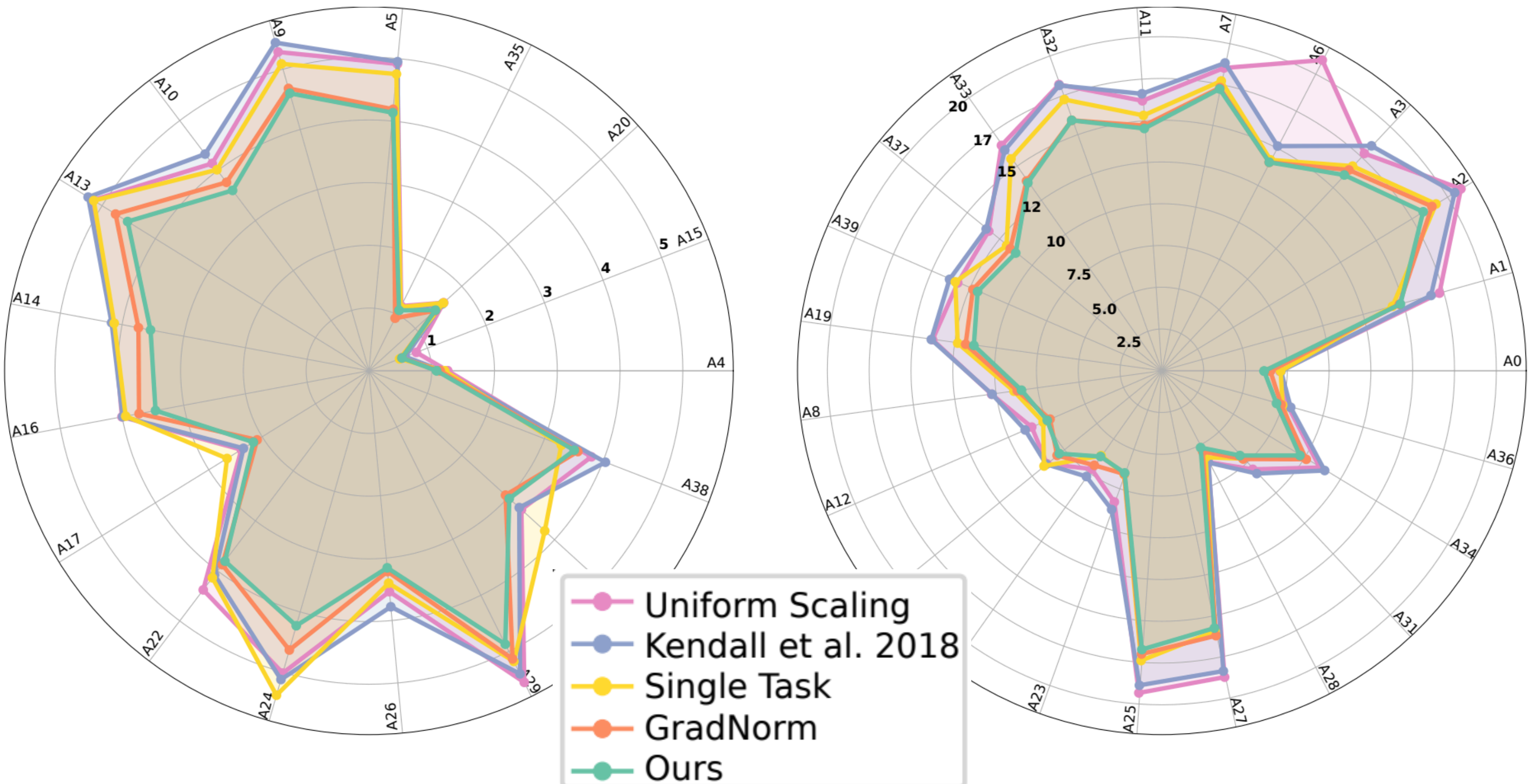
- Then, we can state an upper bound and minimize it efficiently

$$\left\|\sum_{t=1}^{T} \alpha_t \nabla_{\theta_{\mathrm{sh}}} \mathcal{L}_t(\theta_{\mathrm{sh}}, \theta_t)\right\|_2^2 \leq \left\|\frac{\partial Z}{\partial \theta_{\mathrm{sh}}}\right\|_2^2 \left\|\sum_{t=1}^{T} \alpha_t \nabla_Z \mathcal{L}_t(\theta_{\mathrm{sh}}, \theta_t)\right\|_2^2$$

Independent to α_t

Multi-task Learning as Multi-objective Optimization

- 40 binary tasks on CelebA dataset (lower is better)
  - This multi-objective optimization [Sener and Koltun, 2018] outperforms uniform scaling, heuristic weights [Kendall et al., 2018], [Chen et al., 2018]
  - Grid search is not available because there are too many tasks



References

[Pan et al., 2010] A survey on transfer learning, *IEEE Transactions on knowledge and data engineering*, 2010.
link: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5288526

[Weiss et al., 2016] A survey of transfer learning, *Journal of Big Data*, 2016.
link: https://journalofbigdata.springeropen.com/track/pdf/10.1186/s40537-016-0043-6

[Tan et al., 2018] A Survey on Deep Transfer Learning, *arXiv preprint arXiv:1808.01974*, 2018.
link: https://arxiv.org/pdf/1808.01974.pdf

[Yaniv et al., 2014] DeepFace: Closing the Gap to Human-Level Performance in Face Verification, *CVPR*, 2014.
link: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6909616

[Razavian et al., 2014] CNN features off-the-shelf: an astounding baseline for recognition, *CVPR Workshops*, 2014.
link: https://arxiv.org/pdf/1403.6382.pdf

[Hinton et al., 2015] Distilling the knowledge in a neural network, *NIPS workshops*, 2015.
link: https://arxiv.org/pdf/1503.02531.pdf

[Wang et al., 2017] Growing a brain: Fine-tuning by increasing model capacity, *CVPR*, 2017.
link: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8099806

[Ge et al., 2017] Borrowing treasures from the wealthy: Deep transfer learning through selective joint fine-tuning, *CVPR*, 2017.
link: https://arxiv.org/pdf/1702.08690.pdf

[Hendrycks et al., 2019] Using Pre-training Can Improve Model Robustness and Uncertainty , *ICML*, 2019.
link: https://arxiv.org/pdf/1901.09960.pdf

[Andrei et al., 2016] Progressive Neural Networks, *arXiv preprint arXiv:1606.04671*, 2016.
link: https://arxiv.org/pdf/1606.04671.pdf

References

[Kendall et al., 2018] Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics, *CVPR*, 2018.
link: https://arxiv.org/pdf/1705.07115.pdf

[Li et al., 2016] Learning without Forgetting, *ECCV*, 2016.
link: https://arxiv.org/pdf/1606.09282.pdf

[Zagoruyko et al., 2017] Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer, *ICLR*, 2017.
link: https://arxiv.org/pdf/1612.03928.pdf

[Springenberg et al., 2015] Striving for Simplicity: The All Convolutional Net, *ICLR*, 2015.
link: https://arxiv.org/pdf/1412.6806.pdf

[Selvaraju et al., 2016] Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization, *ICCV*, 2017.
link: https://arxiv.org/pdf/1610.02391.pdf

[Czarnecki et al., 2017] Sobolev raining for neural networks, *NIPS*, 2017.
link: https://arxiv.org/pdf/1706.04859.pdf

[Srinivas et al., 2018] Knowledge Transfer with Jacobian Matching, *ICML*, 2018.
link: https://arxiv.org/pdf/1803.00443.pdf

[Cui et al., 2018] Large scale fine-grained categorization and domain specific transfer learning, *CVPR*, 2018.
link: https://arxiv.org/abs/1806.06193

[Ahn et al., 2019] Variational Information Distillation for Knowledge Transfer, *CVPR*, 2019.
link: https://arxiv.org/abs/1904.05835

[Park et al., 2019] Relational Knowledge Distillation, *CVPR*, 2019.
link: https://arxiv.org/abs/1904.05068

References

[Jang and Lee, 2019] Learning What and Where to Transfer, *ICML* 2019.
link: https://arxiv.org/abs/1905.05901

[Tian et al., 2020] Contrastive Representation Distillation, *ICLR*, 2020.
link: https://arxiv.org/abs/1910.10699

[Zhang and Yang, 2017] A Survey on Multi-Task Learning, arxiv, 2017.
link: https://arxiv.org/abs/1707.08114

[Misra et al., 2016] Cross-stitch Networks for Multi-task Learning, CVPR, 2016
link: https://arxiv.org/abs/1604.03539

[Mudrakarta et al., 2019] K for the Price of 1: Parameter-efficient Multi-task and Transfer Learning, ICLR, 2019
link: https://arxiv.org/abs/1810.10703

[Sener and Koltun, 2018] Multi-task Learning as Multi-objective Optimization, NIPS, 2018
link: https://arxiv.org/abs/1810.04650

[Chen et al., 2018] GradNorm: Gradient Normalization for Adaptive Loss Balancing in Deep Multitask Networks, ICML, 2018
link: https://arxiv.org/abs/1711.02257

[Barber and Agakov, 2003] The IM Algorithm: A Variational Approach to Information Maximization, NIPS, 2003
link: http://papers.nips.cc/paper/2410-information-maximization-in-noisy-channels-a-variational-approach.pdf

[Andrychowicz et al., 2016] Learning to learn by gradient descent by gradient descent, NIPS, 2016
link: https://arxiv.org/abs/1606.04474

[Finn et al., 2017] Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks, ICML, 2017
link: https://arxiv.org/abs/1703.03400