Transfer Learning

EE807: Recent Advances in Deep Learning

Lecture 11

Slide made by

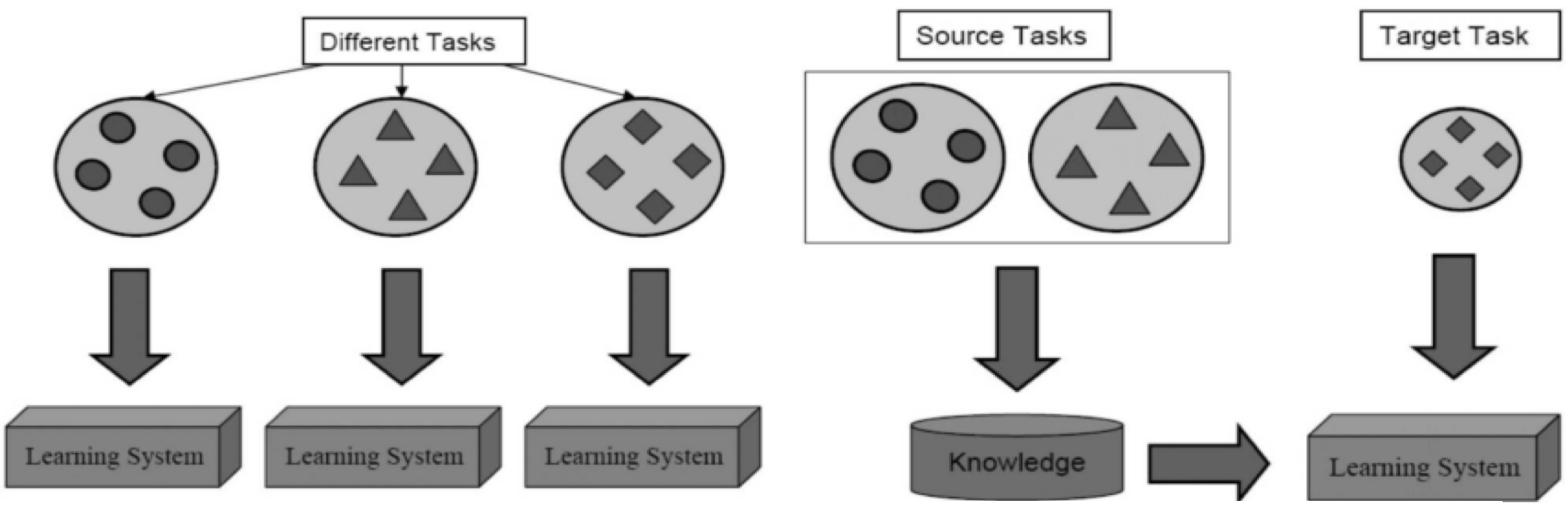
Jongjin Park and Yunhun Jang

KAIST EE

Table of Contents

Table of Contents

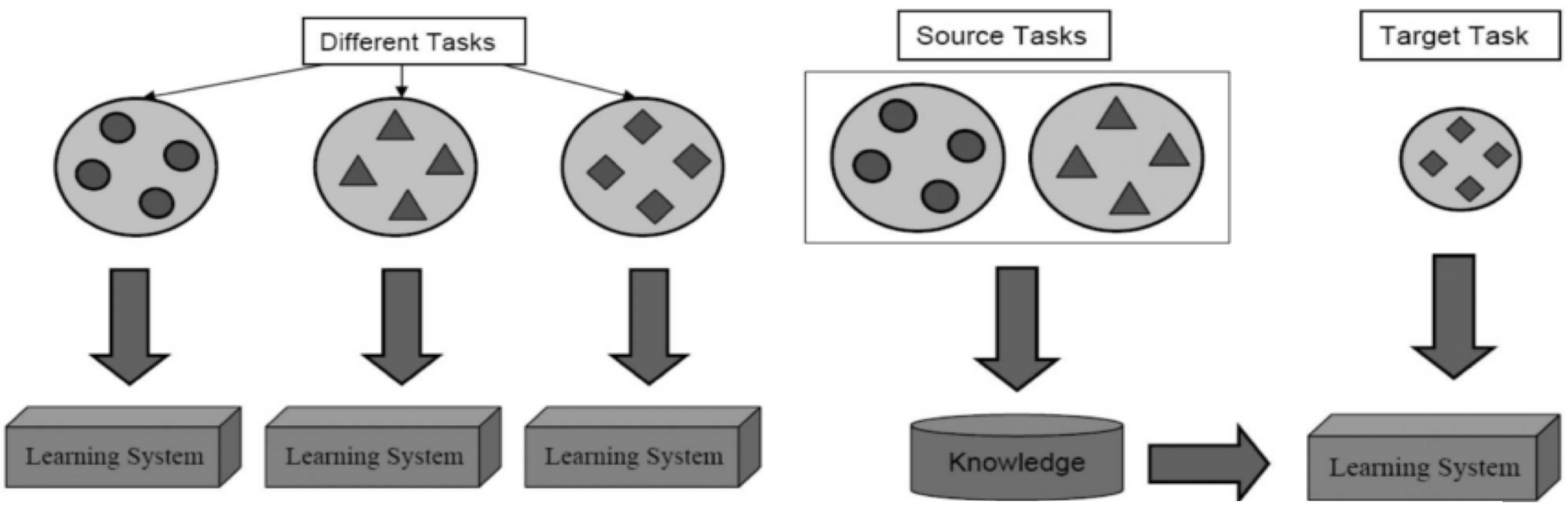
- **Why transfer learning now?**
  - Deep learning shows remarkable success in various fields of artificial intelligence (e.g., object classification, machine translation)
  - But, use (VERY) large labeled dataset
    - How about novel/new tasks? What if we do not have labeled datasets?
- Transfer learning aims to **extract the knowledge** from one or more **source tasks** and applies the knowledge to a **target task**



Learning process of traditional machine learning   Learning process of transfer learning

* source :[Pan et al., 2010]

- Definition [Pan et al., 2010]
  - A ***domain*** is defined as a pair $\mathcal{D} = \{\mathcal{X}, P(X)\}$, which consists a feature space $\mathcal{X}$, and a marginal distribution $P(X)$ over the feature space.
  - A ***task*** is defined as a pair $\mathcal{T} = \{\mathcal{Y}, P(Y \mid X)\}$, which consists a label space $\mathcal{Y}$, and a conditional distribution $P(Y \mid X)$.
  - Given
    - A source domain $\mathcal{D}_S$ and learning task $\mathcal{T}_S$
    - 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 $\mathcal{T}_S \neq \mathcal{T}_T$.



Learning process of traditional machine learning

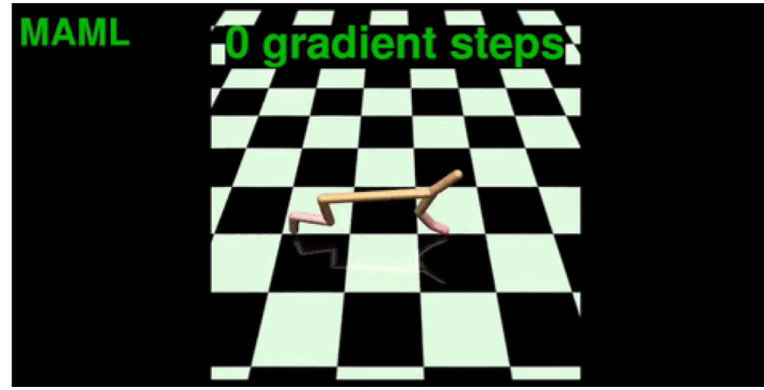
Learning process of transfer learning

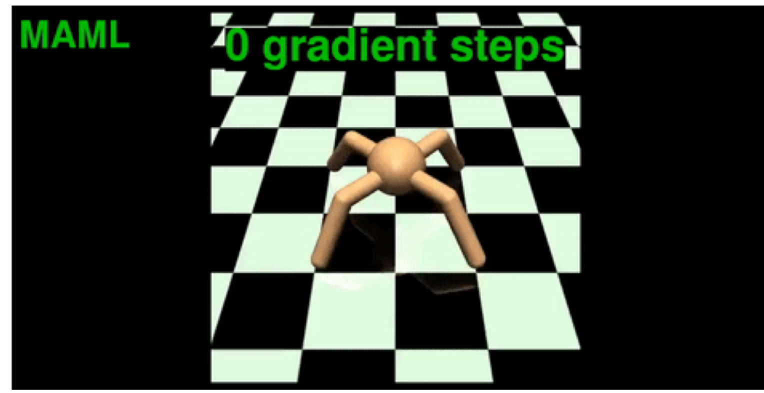
* source :[Pan et al., 2010]

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

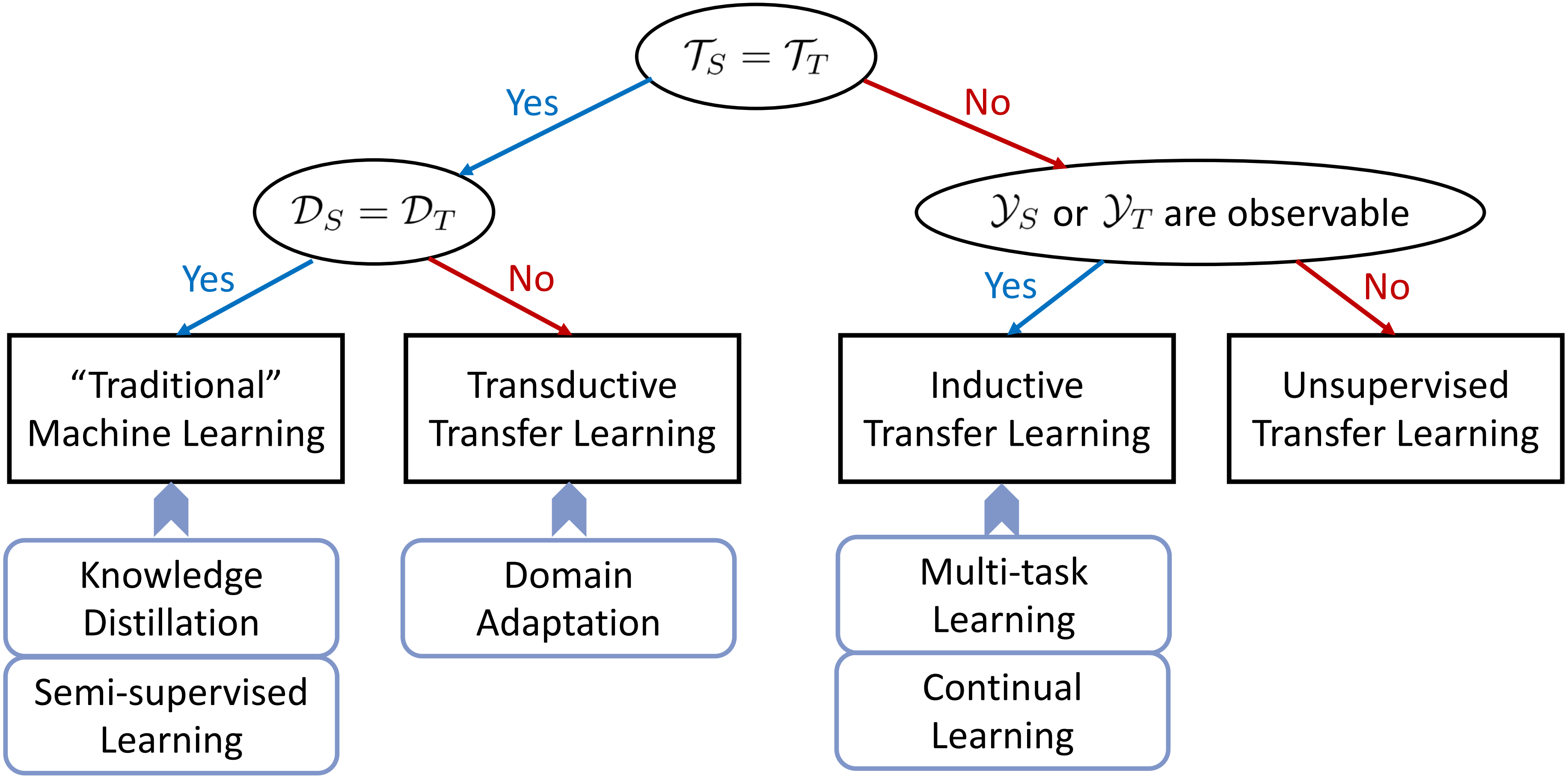




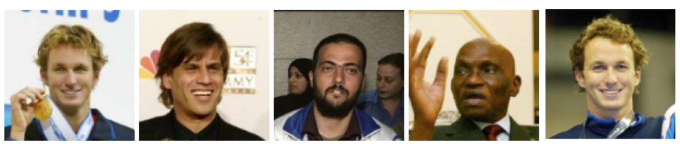
Simulated robots learn new movements from get transfer from previous learned task
(Top): from forward movements, learn backward move
(Bottom): learn faster movements from slow movements

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

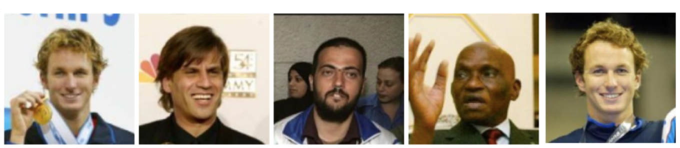
- Recap.
  - ***Transfer learning*** setting is $\mathcal{D}_S \neq \mathcal{D}_T$, or $\mathcal{T}_S \neq \mathcal{T}_T$.
  - There are various scenarios depend on the detail settings [Pan et al., 2010].



- 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)

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

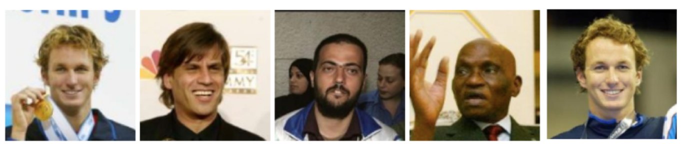
- Labels to predict are same but input data samples are different
  - Since tasks are same, by learning the features *invariant* to source and target domains, a target model can perform well
  - In many cases, target domain datasets do not have sufficient labels
    - By learning domain invariant features,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

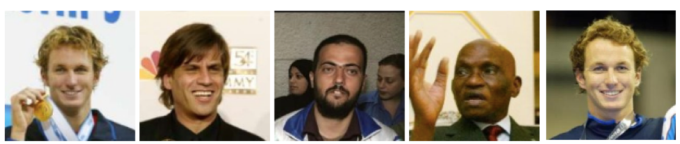
0-9 digits classification
(e.g., 0, 4, 8, 5, 3, 9)

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

- 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)

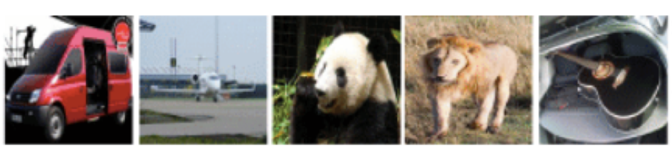


Person recognition
(e.g., John, Aaron, Adam, Will, John)

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

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

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



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

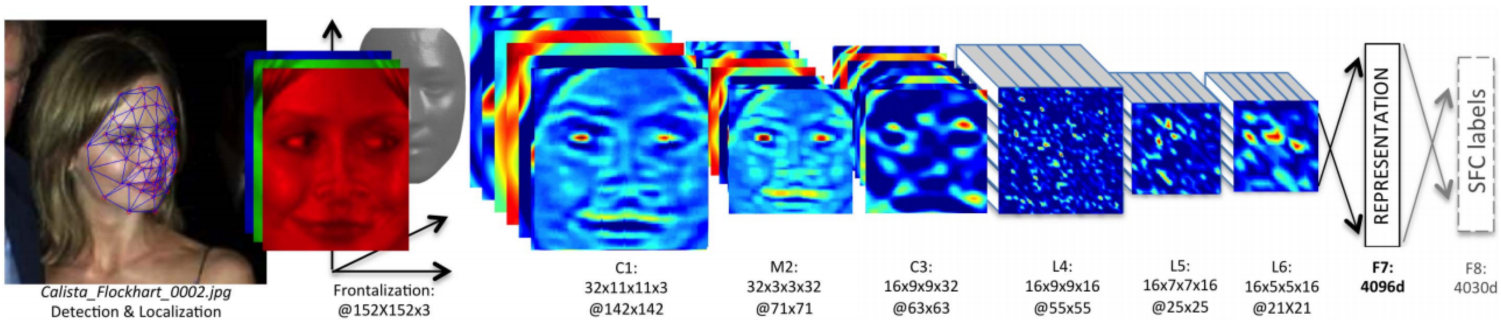
$\mathcal{D}_S \neq \mathcal{D}_T$

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

Table of Contents

Fine-tuning Approach

- 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)

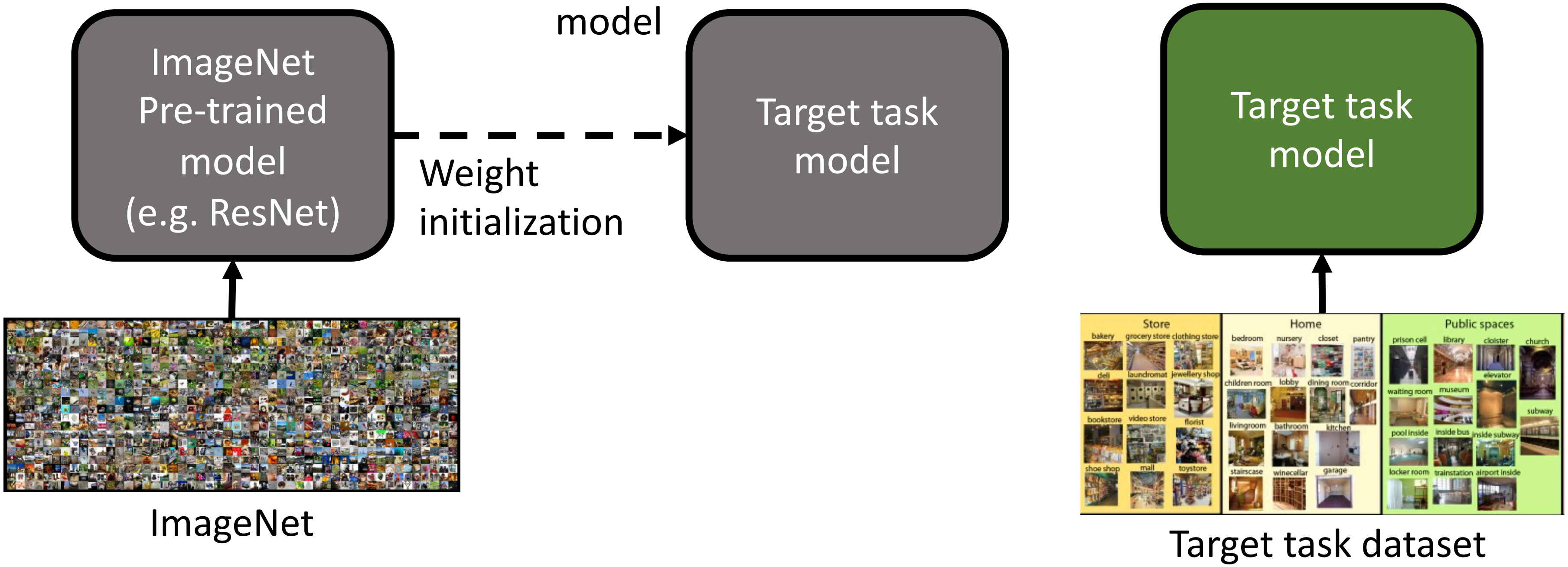
* source: [Yaniv et al., 2014]

Fine-tuning Approach

(a) Train large model on ImageNet

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

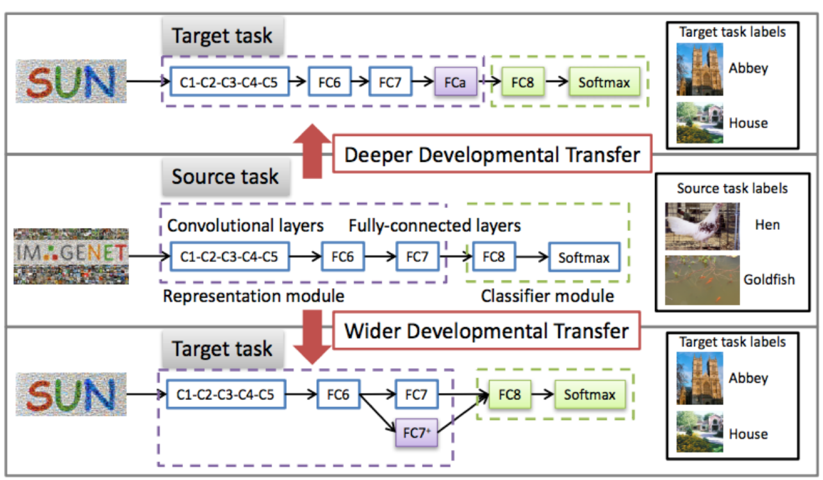
(c) Fine-tuning the weights for new task

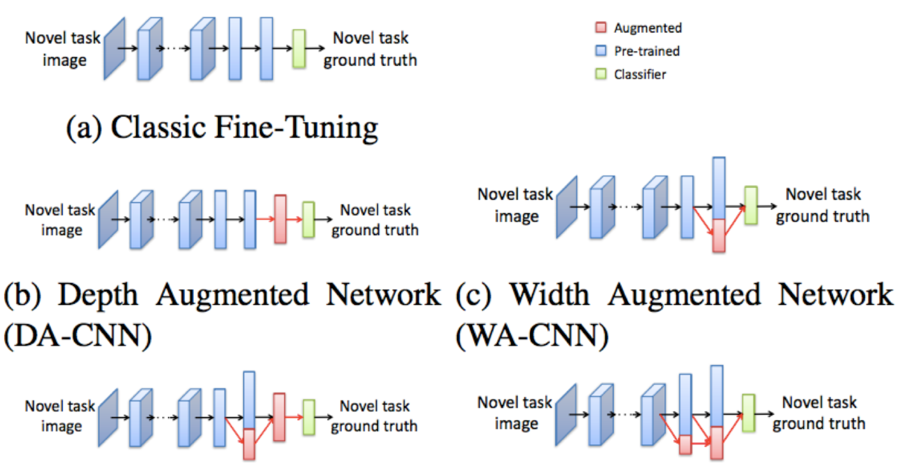


- Assumptions for fine-tuning approaches
  - **Features/Parameters** learned from some task are useful for **another tasks**
    - True in many artificial intelligence tasks (e.g. lower-level features of images such as edge)
- When do they **fail** to work
  - When **dataset** of source and target tasks 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





(a) Classic Fine-Tuning

(b) Depth Augmented Network (DA-CNN)

(c) Width Augmented Network (WA-CNN)

(d) Jointly Depth and Width Augmented Network (DWA-CNN)

(e) Recursively Width Augmented Network (WWA-CNN)

* source: [Wang et al., 2017]

Experimental Results

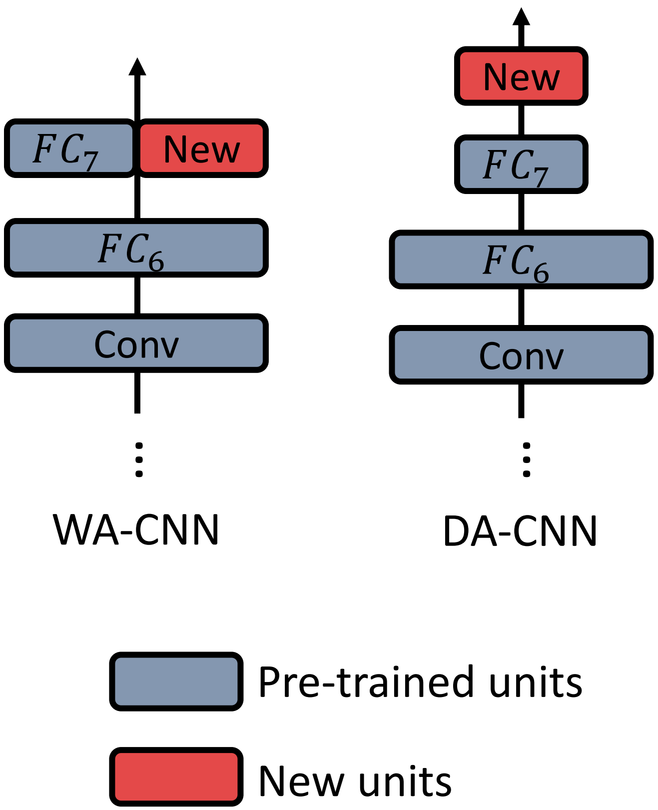
- Evaluated on MIT-67, 102 Flowers, CUB200-2011, Stanford-40 with ImageNet pre-trained AlexNet
- Outperform most of task customized CNN or other multi-task learning methods
- Drawbacks:
  - Did not apply on architecture like ResNet (model without fully-connected layers)
  - Only augment the layers for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

Experimental Results

- 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

$$\hat{\boldsymbol{h}}^k = \underbrace{\gamma}_{\text{Scaling}} \boldsymbol{h}^k / \underbrace{\left\|\boldsymbol{h}^k\right\|_2}_{\text{Normalization}}$$

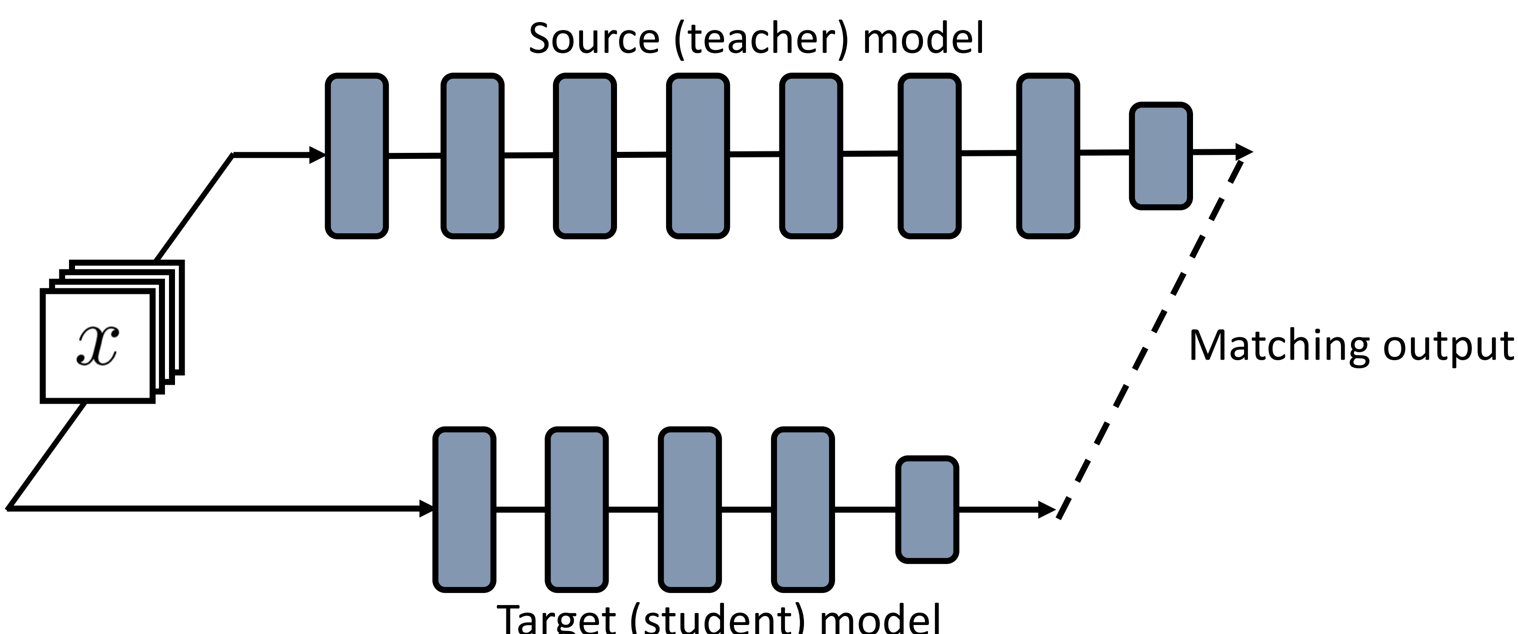


Method	Scaling	New	FC_7-new	FC_6-new	All
Fine-tuning CNN	-	53.63	54.75	54.29	55.93
DA-CNN	w/o (rand)	**53.82**	**56.47**	56.25	57.21
	w/	53.51	56.15	**57.14**	**58.07**
WA-CNN	w/o (rand)	53.78	54.66	49.72	51.34
	w/o (copy+rand)	53.62	54.35	53.70	55.31
	w/	**56.81**	**56.99**	**57.84**	**58.95**

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

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)



Knowledge Distillation: Matching Output of Source and Target Model

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

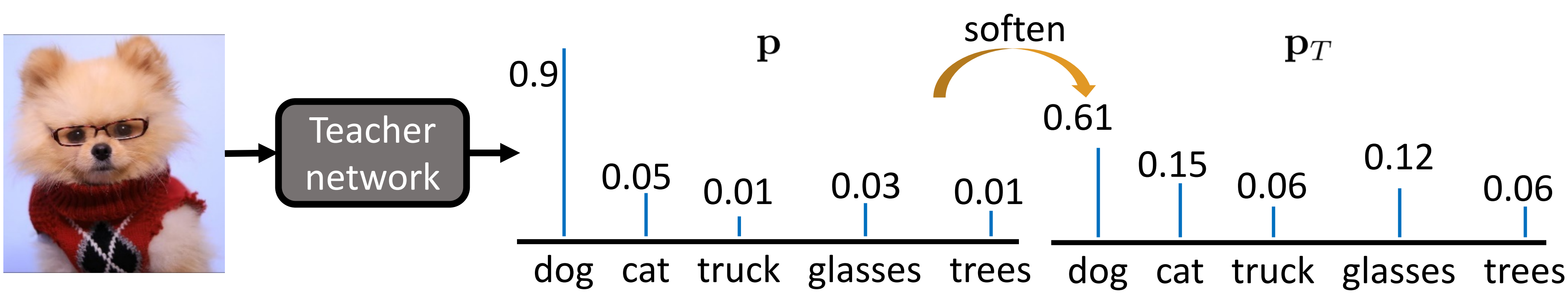
$$q_{i,T} = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)}$$

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)



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,

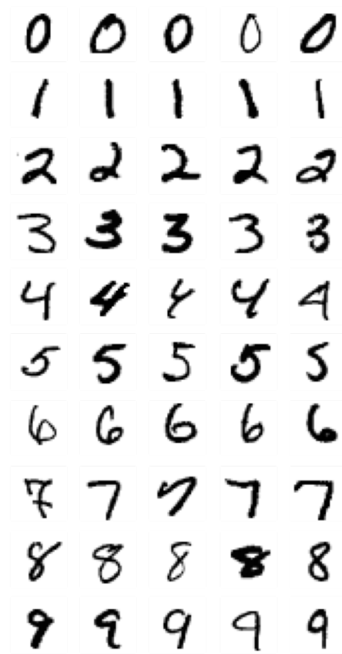
$$\frac{\partial C}{\partial z_i} \approx \frac{1}{T}\left(\frac{1+z_i/T}{N+\sum_j z_j/T} - \frac{1+v_i/T}{N+\sum_j v_j/T}\right)$$

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

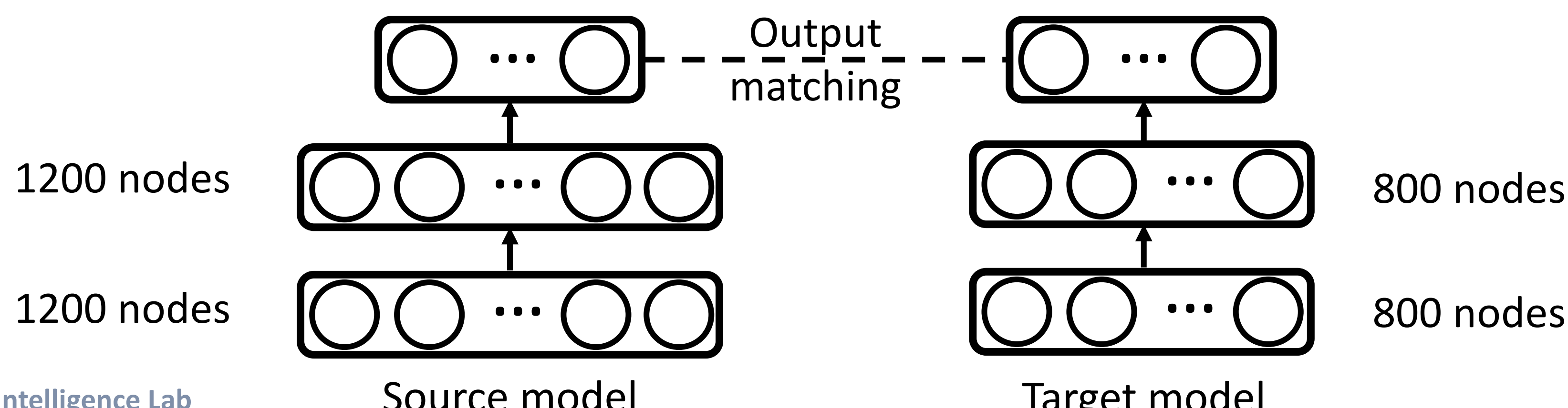
$$\frac{\partial C}{\partial z_i} \approx \frac{1}{NT^2}(z_i - v_i) = \underbrace{\boxed{\frac{1}{NT^2}}}_{\text{scaling}}\frac{\partial}{\partial z_i}\left(\boxed{\frac{1}{2}(z_i - v_i)^2}\right)$$

- **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



Beyond Knowledge Distillation

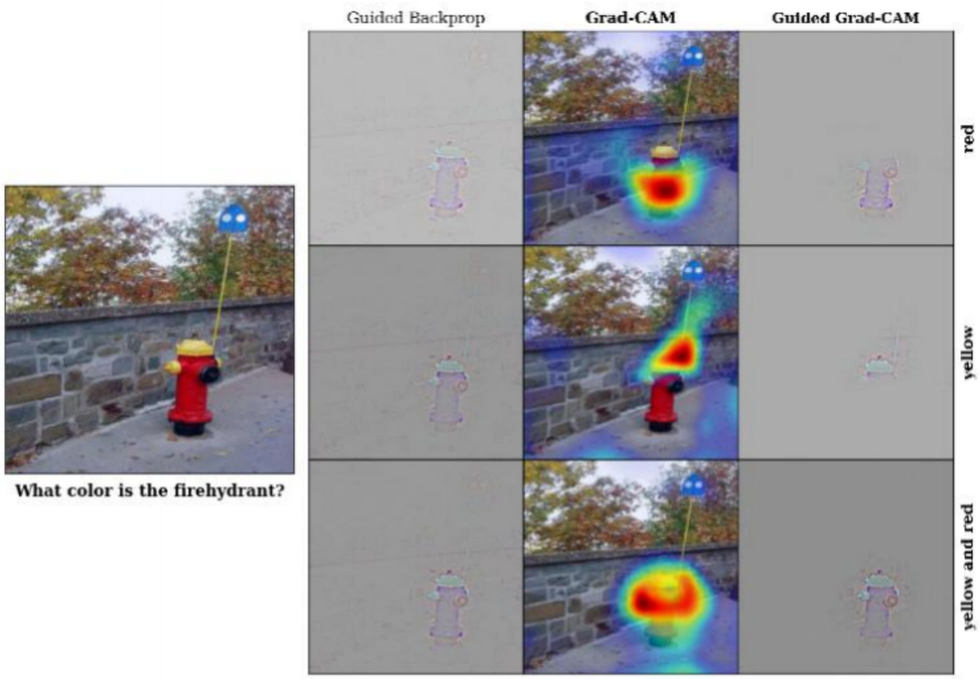
- Smaller target models get advantages by following larger source models

- 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

Table of Contents

- 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.

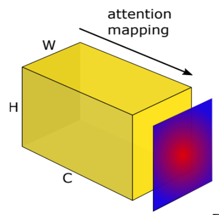


Visualization of VQA model.

* 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

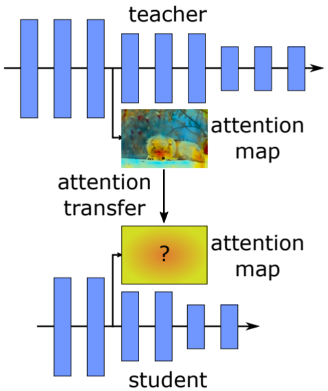
$$F(A_{h,w}) = \sum_{c=1}^{C} |A_{c,h,w}|^p$$

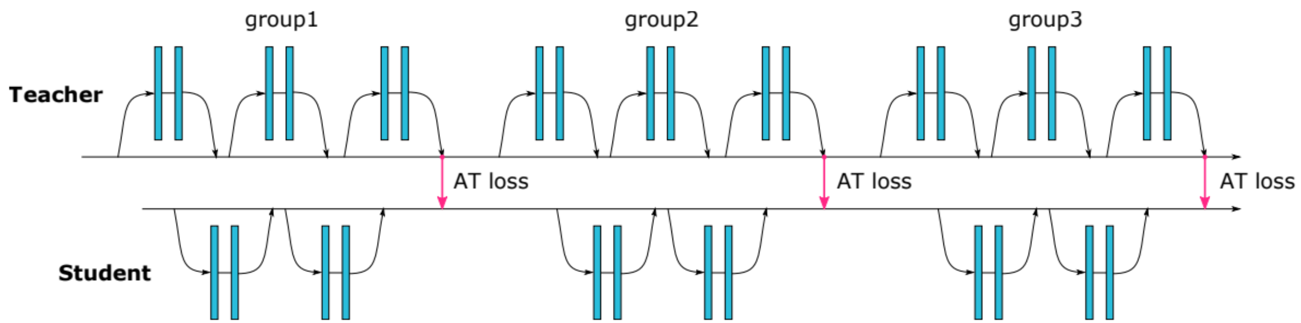


  - $p > 1$, feature activation $A_{c,h,w} \in \mathbb{R}^{C \times H \times W}$ (C channels, spatial size $H \times W$)
  - Train the original loss with the attention map matching regularization term

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

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]

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

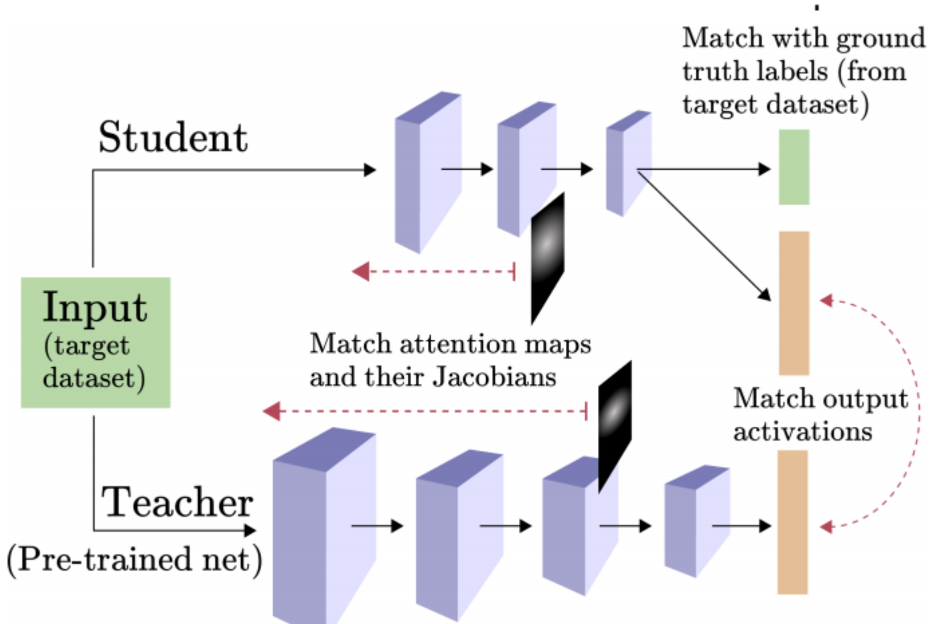
CIFAR-10 experiments. **AT**: attention transfer, **F-ActT**: full activation transfer, **KD**: knowledge distillation **AT+KD**: applying AT and KD at the same time. AT+KD is best in most cases (for student networks)

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.

Jacobian Matching

- 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).



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

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

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

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]

Experimental Results

- 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

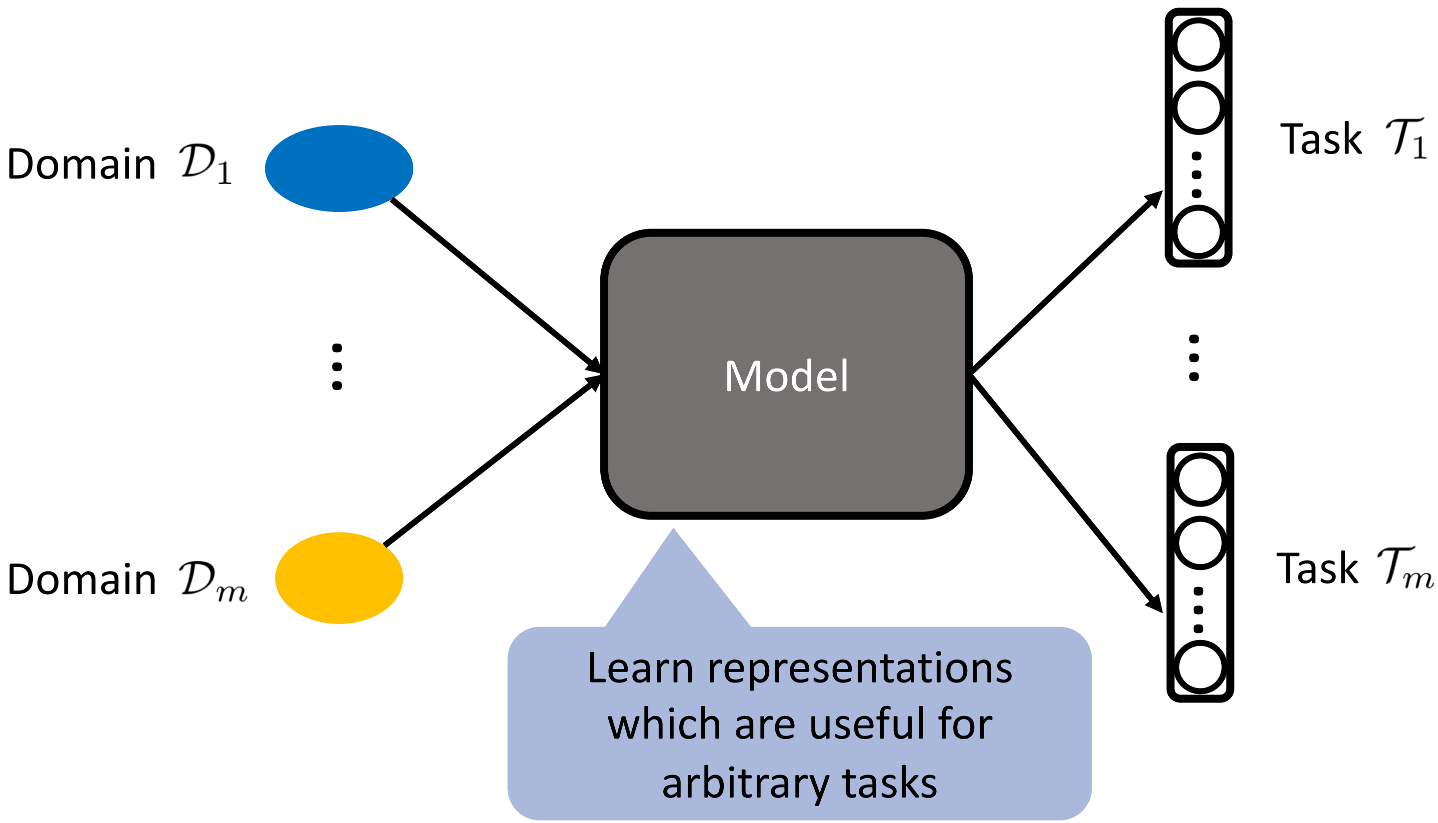
- Matching Jacobians improves performance of all case of transfer learning.
- None of the methods match the oracle performance of pre-trained model.

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**

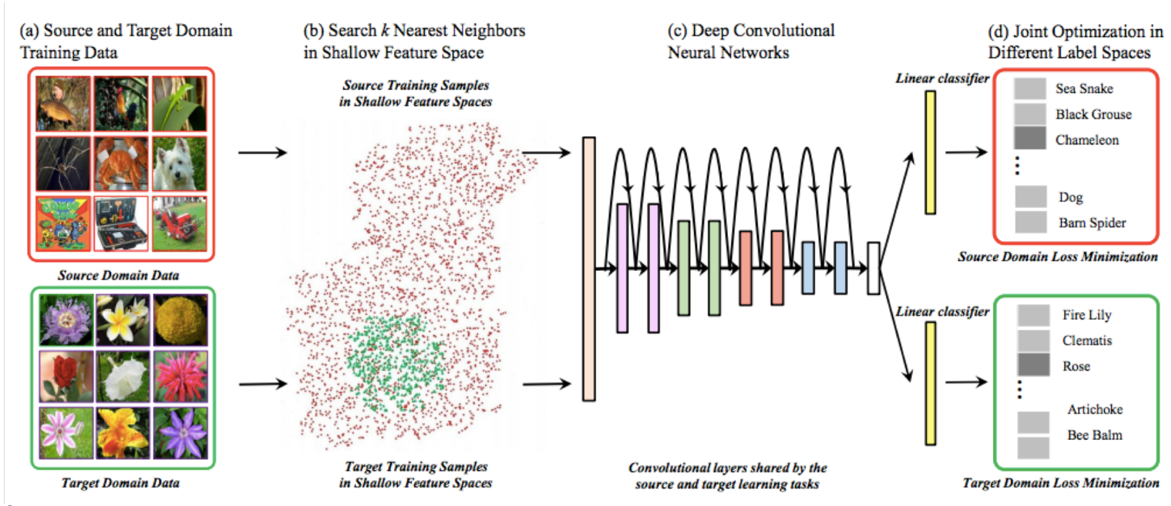
Table of Contents

- Multi-task learning aims to improve all tasks *simultaneously* by combining the common knowledge from all tasks.
  - Transfer knowledge by **jointly learning** all the tasks
  - Target tasks with small dataset can take advantages from jointly learning the tasks



Multi-Task Learning for Transfer

- Joint fine-tuning with an useful subset of source images [Ge et al., 2017]
  - Convolutional layers are shared
  - Output layers are separated for source and target tasks
- **Identify and use a subset** of training images from the original source learning task
  - Choose source dataset samples whose low-level characteristics are similar to those from the target learning task
- Similar image search
  - Using shallow features (e.g. Gabor filter or 1st and 2nd layers features of AlexNet)
  - Make image descriptor (histogram) using obtained features
  - Find similar image using k nearest neighbors



* source: [Ge et al., 2017]

Experimental Results

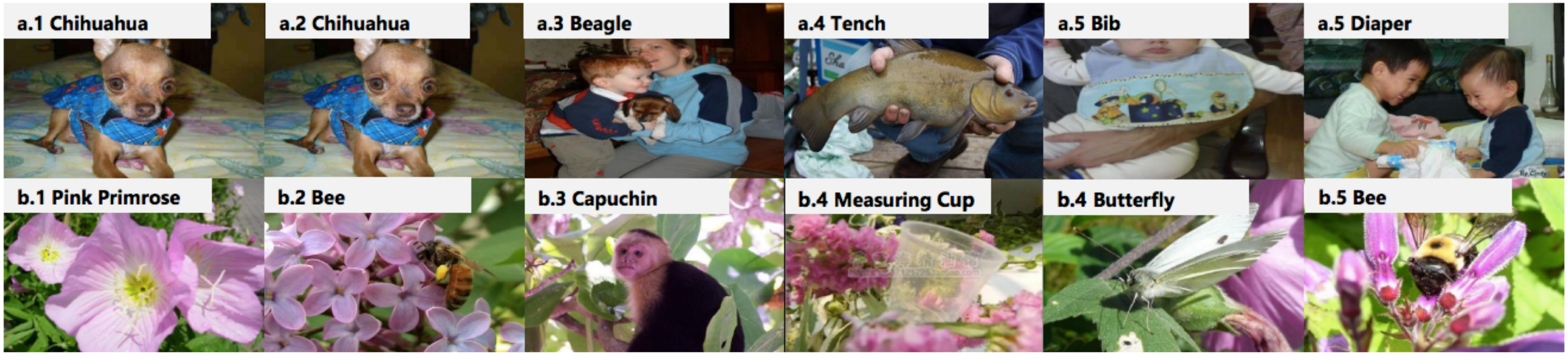


Figure. Similar Images for Stanford Dogs 120 dataset (1st row) and Oxford Flowers 102 (2nd row). First columns are images from target dataset, and others from ImageNet

Method	mean Acc(%)
HAR-CNN [44]	49.4
Local Alignment [9]	57.0
Multi scale metric learning [32]	70.3
MagNet [35]	75.1
Web Data + Original Data [21]	85.9
Training from scratch using target domain only	53.8
Selective joint training from scratch	83.4
Fine-tuning w/o source domain	80.4
Joint fine-tuning with all source samples	85.6
Selective joint FT with random source samples	85.5
Selective joint FT w/o iterative NN retrieval	88.3
Selective joint FT with Gabor filter bank	87.5
Selective joint fine-tuning	90.2
Selective joint FT with Model Fusion	**90.3**

Method	mean Acc(%)
MPP [47]	91.3
Multi-model Feature Concat [1]	91.3
MagNet [35]	91.4
VGG-19 + GoogleNet + AlexNet [20]	94.5
Training from scratch using target domain only	58.2
Selective joint training from scratch	80.6
Fine-tuning w/o source domain	92.3
Joint fine-tuning with all source samples	93.4
Selective joint FT with random source samples	93.2
Selective joint FT w/o iterative NN retrieval	94.2
Selective joint FT with Gabor filter bank	93.8
Selective joint fine-tuning	94.7
Selective joint FT with model fusion	**95.8**
VGG-19 + Part Constellation Model [38]	95.3
Selective joint FT with val set	**97.0**

Table. (Left) Results of Stanford Dogs 120, (Right) Results of Oxford Flowers 102

* source: [Ge et al., 2017]

- 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 $\sigma \rightarrow \infty$.

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

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

  - 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

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

Experimental Results

- 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)

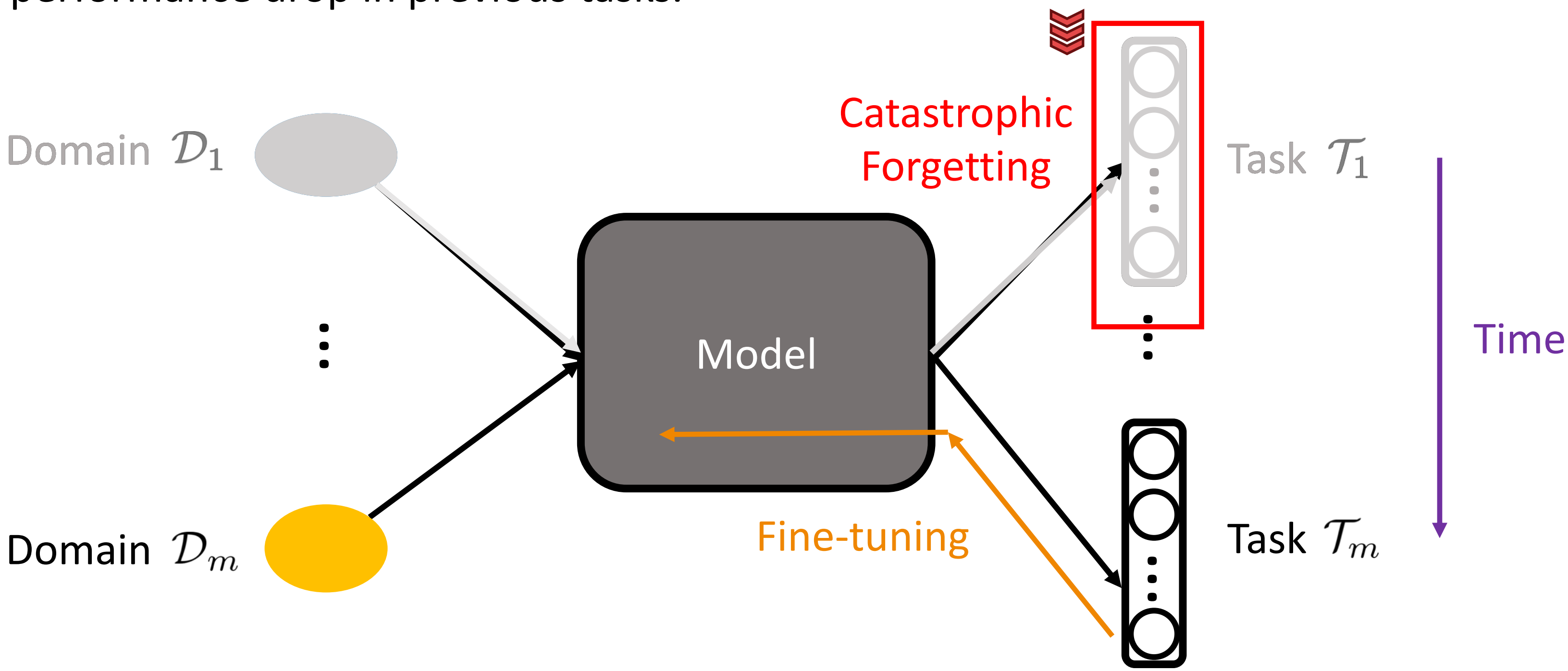
Approx. optimal weights are found by grid search.

Loss	Task Weights Seg.	Inst.	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]

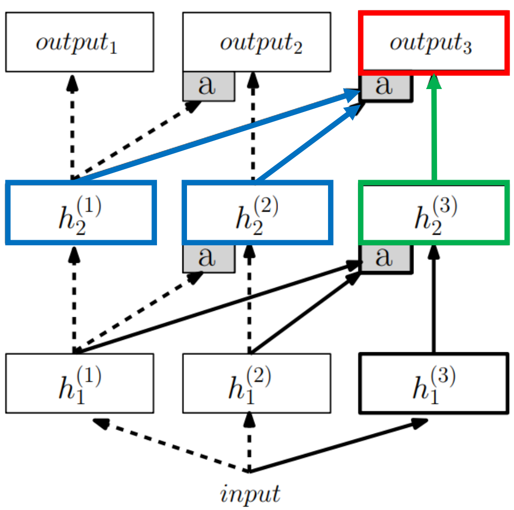
Table of Contents

- One distinctive ability of humans is to *continually learn* new skills and accumulate knowledge throughout the lifetime.
- Continual learning (a.k.a. lifelong learning) aims to improve the new task by combining the knowledge from the previous *sequential* tasks.
  - At most cases, the model has a limitation of accessing previous data due to memory and time cost.
  - Simple fine-tuning method occurs *catastrophic forgetting*, which is a significant performance drop in previous tasks.



Progressive Neural Networks

- *Progressive networks* integrate subnetworks into the model architecture. [Andrei et al. 2016]
  - Catastrophic forgetting is prevented by instantiating a new neural network (a column) for each task being solved, while transfer is enabled via lateral connections.
  - The parameters for previous tasks are *"frozen"* and a new column is added(with random initialization), where layer $h_i^{(k)}$ receives input from $h_{i-1}^{(1)}$ to $h_{i-1}^{(k-1)}$.
- In practice, the authors augmented the progressive network layer with non-linear lateral connections which they call *adapters*.
  - They replace the linear lateral connection with a single hidden layer MLP
  - Before feeding the MLP, multiply the hidden layer by a learned scalar $\alpha_{i-1}^{(j)}$.



$$h_i^{(k)} = f\left(W_i^{(k)} h_{i-1}^{(k)} + \sum_{j<k} U_i^{(k:j)} h_{i-1}^{(j)}\right)$$

Using adapters

$$h_i^{(k)} = \sigma\left(W_i^{(k)} h_{i-1}^{(k)} + U_i^{(k:j)} \sigma(V_i^{(k:j)} \alpha_{i-1}^{(<k)} \odot h_{i-1}^{(<k)})\right)$$

where $h_{i-1}^{(<k)} = [h_{i-1}^{(1)} \dots h_{i-1}^{(j)} \dots h_{i-1}^{(k-1)}] \in \mathbb{R}^{n_{i-1}^{(<k)}}$

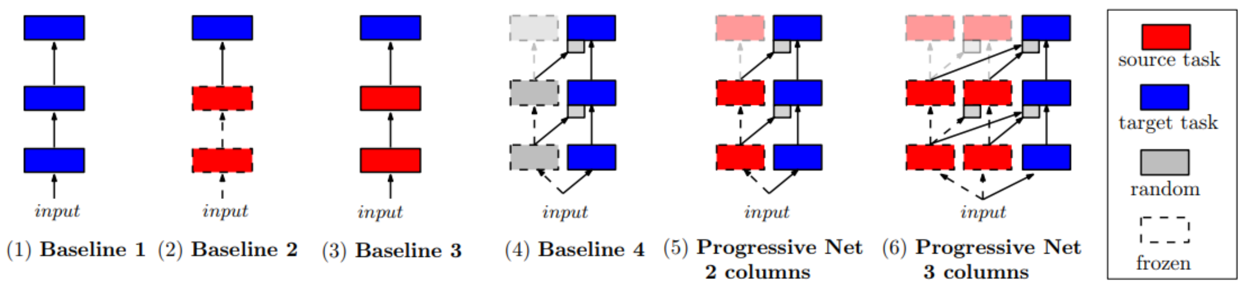
$\alpha_{i-1}^{(<k)} = [\alpha_{i-1}^{(1)} \dots \alpha_{i-1}^{(j)} \dots \alpha_{i-1}^{(k-1)}]$

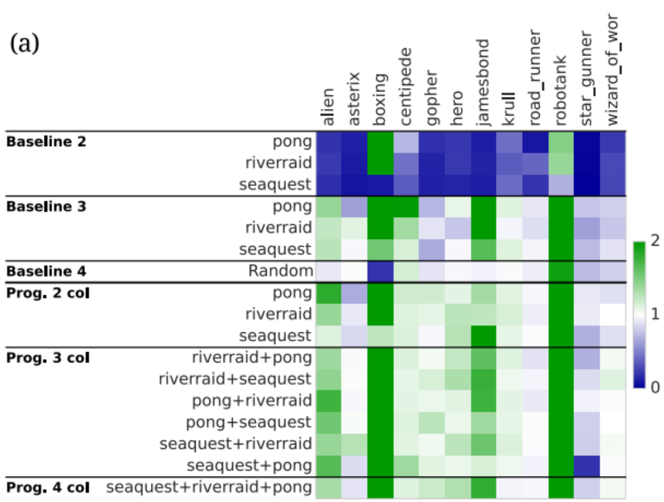
$W_i^{(k)} \in \mathbb{R}^{n_i \times n_{i-1}}$ weight matrix

$U_i^{(k:j)} \in \mathbb{R}^{n_i \times n_j}$ lateral connections

$V_i^{(k:j)} \in \mathbb{R}^{n_i \times n_{i-1}^{(<k)}}$ projection matrix

- Evaluated in reinforcement learning tasks.
- They investigate feature transfer between randomly selected Atari games.
  - They trained single columns on three source games (Pong, River Raid, and Seaquest), and then trained a different subset of randomly selected target games.



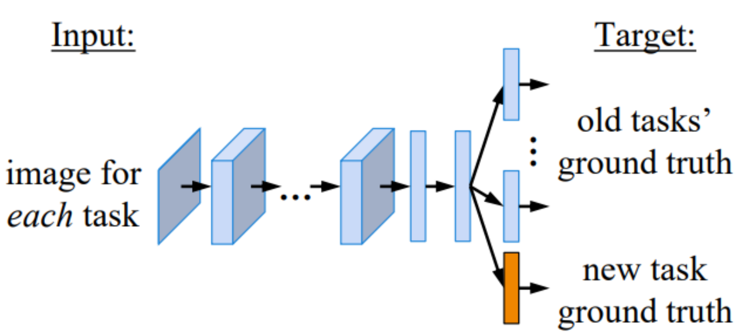


Transfer scores

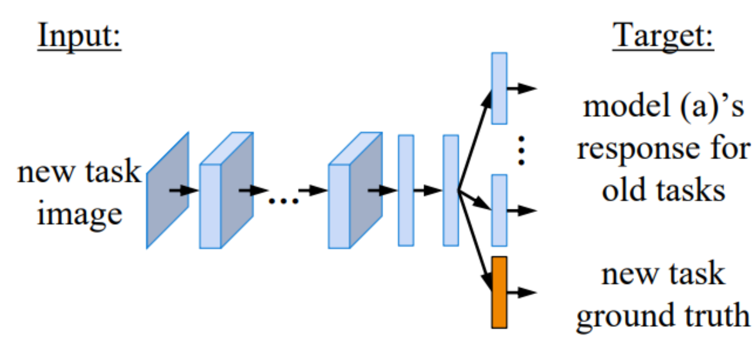
The transfer score is defined as the relative performance compared with a baseline 1.

- Distillation from old tasks' knowledge [Li et al. ,2016]
  - Similar to joint training, the objective is each task's output, but it uses only examples for the new task.
  - In LwF, we use pretrained model's response for old task, instead of the ground truth labels of old data.
  - LwF is computationally efficient.
    - Training time is faster than joint training because we need only one forward-propagation to get multiple outputs.
  - LwF is also memory-efficient.
    - We don't have to store previous data for old task.

Joint Training



Learning without Forgetting



* source: [Li et al., 2016]

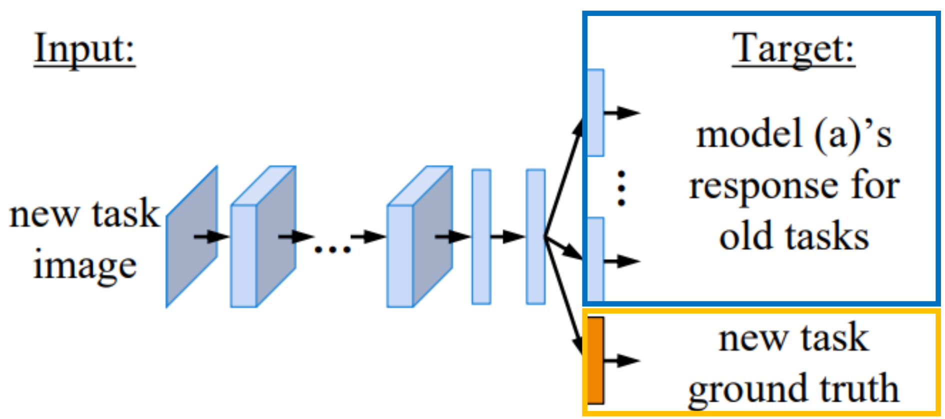
- For each original task, we want to the output to be close to the recorded output from the original network.
  - How to transfer knowledge from softened label effectively?
  - **Knowledge Distillation** $\mathcal{L}_{\text{old}}(y, \hat{y}) = \mathcal{L}_{\text{ce}}(y', \hat{y}')$

    where $y'^{(i)} = \frac{(y^{(i)})^{1/T}}{\sum_j (y^{(j)})^{1/T}}, \quad y'^{(i)} = \frac{(\hat{y}^{(i)})^{1/T}}{\sum_j (\hat{y}^{(j)})^{1/T}}$

    which are softened outputs of old tasks with temperature T
- For the new task, we use simple cross-entropy loss (in classification setting).

$$\mathcal{L}_{\text{new}}(y, \hat{y}) = \mathcal{L}_{\text{ce}}(y, \hat{y})$$

- Total loss is $\mathcal{L} = \lambda \mathcal{L}_{\text{old}}(y, \hat{y}) + \mathcal{L}_{\text{new}}(y, \hat{y}) + \mathcal{R}(\theta)$
  where $\mathcal{R}$ is a regularization term, and λ is a weight parameter.



Distillation from old tasks

* source: [Li et al., 2016]

- LwF outperforms feature extraction and surprisingly, it sometimes outperforms fine-tuning on the new task .
- This method also generally performs better than other baselines.

(a) Using AlexNet structure (validation performance for ImageNet/Places365/VOC)

	ImageNet→VOC		ImageNet→CUB		ImageNet→Scenes		Places365→VOC		Places365→CUB		Places365→Scenes		ImageNet→MNIST	
	old	new	old	new	old	new	old	new	old	new	old	new	old	new
LwF (ours)	56.2	76.1	54.7	57.7	55.9	64.5	50.6	70.2	47.9	34.8	50.9	75.2	49.8	99.3
Fine-tuning	-0.9	-0.3	-3.8	-0.7	-2.0	-0.8	-2.2	0.1	-4.6	1.0	-2.1	-1.7	-2.8	0.0
LFL	0.0	-0.4	-1.9	-2.6	-0.3	-0.9	0.2	-0.7	0.7	-1.7	-0.2	-0.5	-2.9	-0.6
Fine-tune FC	0.5	-0.7	0.2	-3.9	0.6	-2.1	0.5	-1.3	1.8	-4.9	0.3	-1.1	7.0	-0.2
Feat. Extraction	0.8	-0.5	2.3	-5.2	1.2	-3.3	1.1	-1.4	3.8	-12.3	0.8	-1.7	7.3	-0.8
Joint Training	0.7	-0.2	0.6	-1.1	0.5	-0.6	0.7	-0.0	2.3	1.5	0.3	-0.3	7.2	-0.0

Fine-tuning: fine-tune full networks, **LFL**: Less Forgetting Learning(similar previous method), **Fine-tune FC**: freeze the convolutional layers to prevent overfitting, **Feat.Extraction**: similar as fine-tune fc, but last fc layers can be more than one. **Joint Training**: jointly train multiple tasks. (upper bound)

(b) Test set performance

	Places365→VOC	
	old	new
LwF (ours)	50.6	73.7
Fine-tuning	-2.1	0.1
Feat. Extraction	1.3	-2.3
Joint Training	0.9	-0.1

(c) Using VGGnet structure

	ImageNet→CUB		ImageNet→Scenes	
	old	new	old	new
LwF (ours)	60.6	72.5	66.8	74.9
Fine-tuning	-9.9	0.6	-4.1	-0.3
LFL	0.3	-2.8	-0.0	-2.1
Fine-tune FC	3.2	-6.7	1.4	-2.4
Feat. Extraction	8.2	-8.6	1.9	-5.1
Joint Training	8.0	2.5	4.1	1.5

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

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

[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

References

[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