# Learning What and Where to Transfer

**Yunhun Jang** [*1,2] **Hankook Lee** [*1] **Sung Ju Hwang** [3,4,5] **Jinwoo Shin** [1,4,5]

[1]School of Electrical Engineering, KAIST, Korea
[2]OMNIOUS, Korea
[3]School of Computing, KAIST, Korea
[4]Graduate School of AI, KAIST, Korea
[5]AITRICS, Korea.

ICML 2019

# Content

- Introduction
- Method
- Experiment

# Content

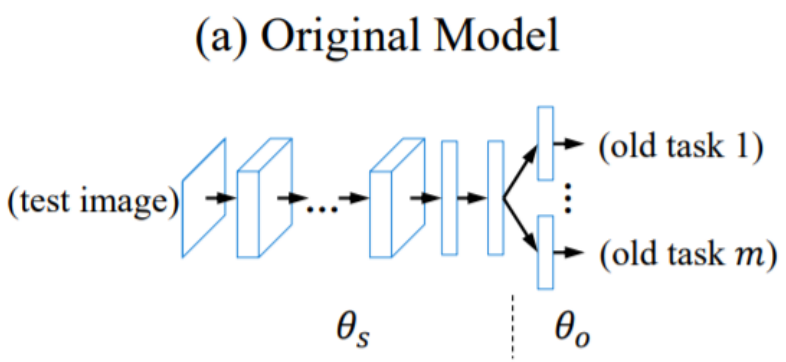
- Introduction
- Method
- Experiment

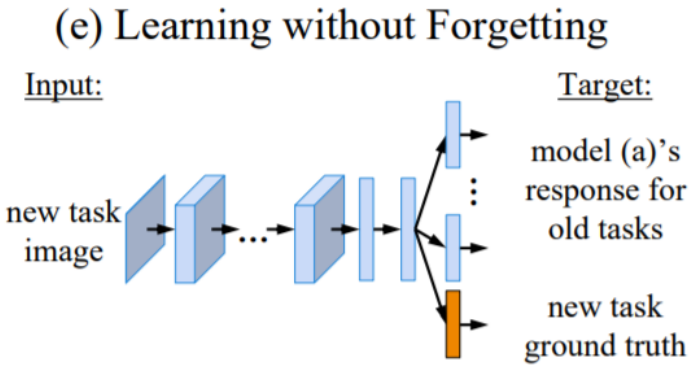
# Introduction

- It is expensive to collect a sufficient amount of labeled samples for DNN.
- Transfer Learning: transfer knowledge from a known source task to a new target task.
- Pre-training with fine-tune:
  - First train a source DNN with a large dataset
  - Then use the learned weights as an initialization to train a target DNN
- Fine-tune has problems:
  - Source and target tasks are semantically distant
  - Cannot be used when network architectures for the source and target tasks largely differ

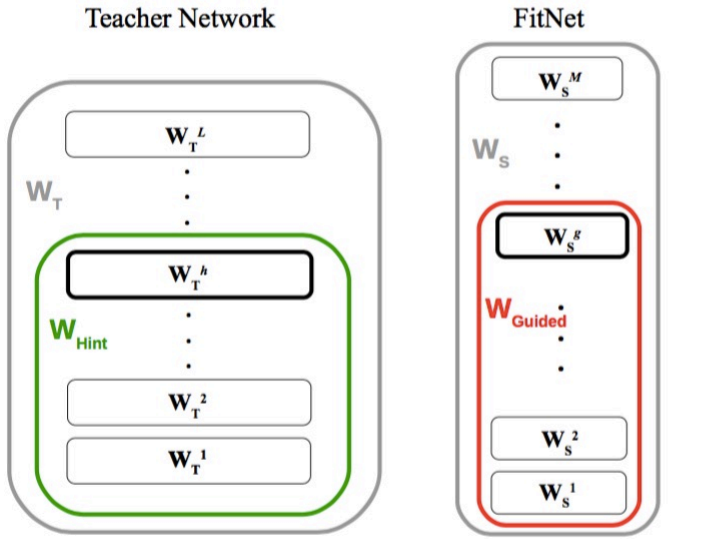
# Related Work

- Learning without forgetting (LwF)

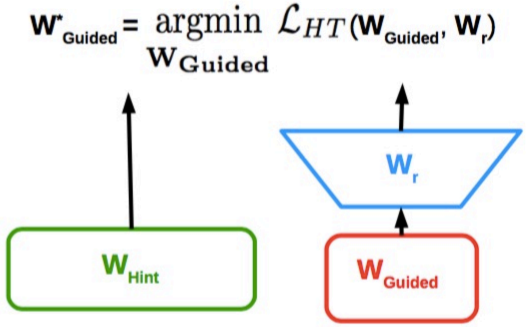



- FitNet




(a) Teacher and Student Networks




(b) Hints Training

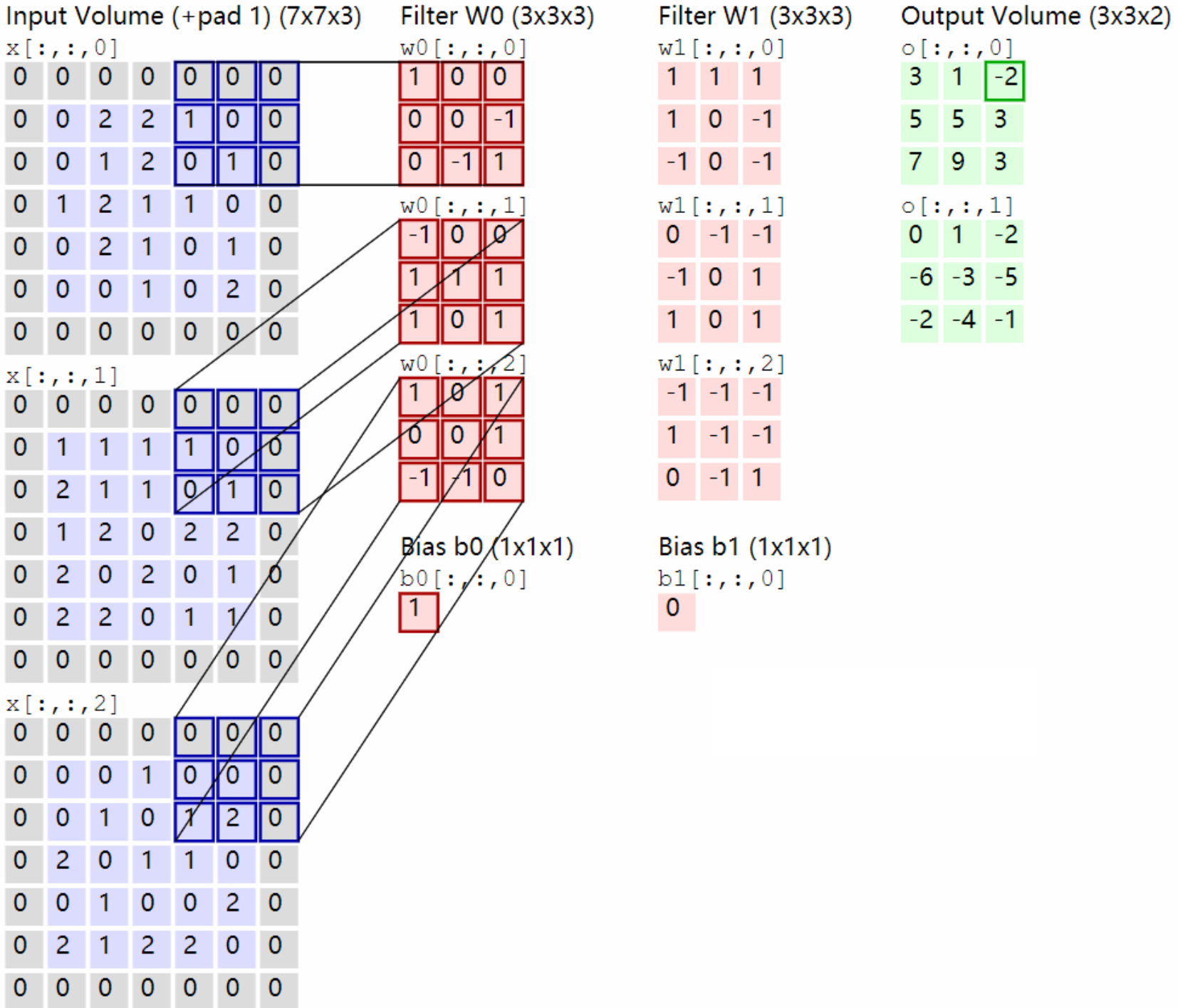
# Motivation

- ✓ Transfer knowledge between heterogeneous source and target tasks/architectures
- × Have no mechanism to identify which source information to transfer, between which layers of the networks

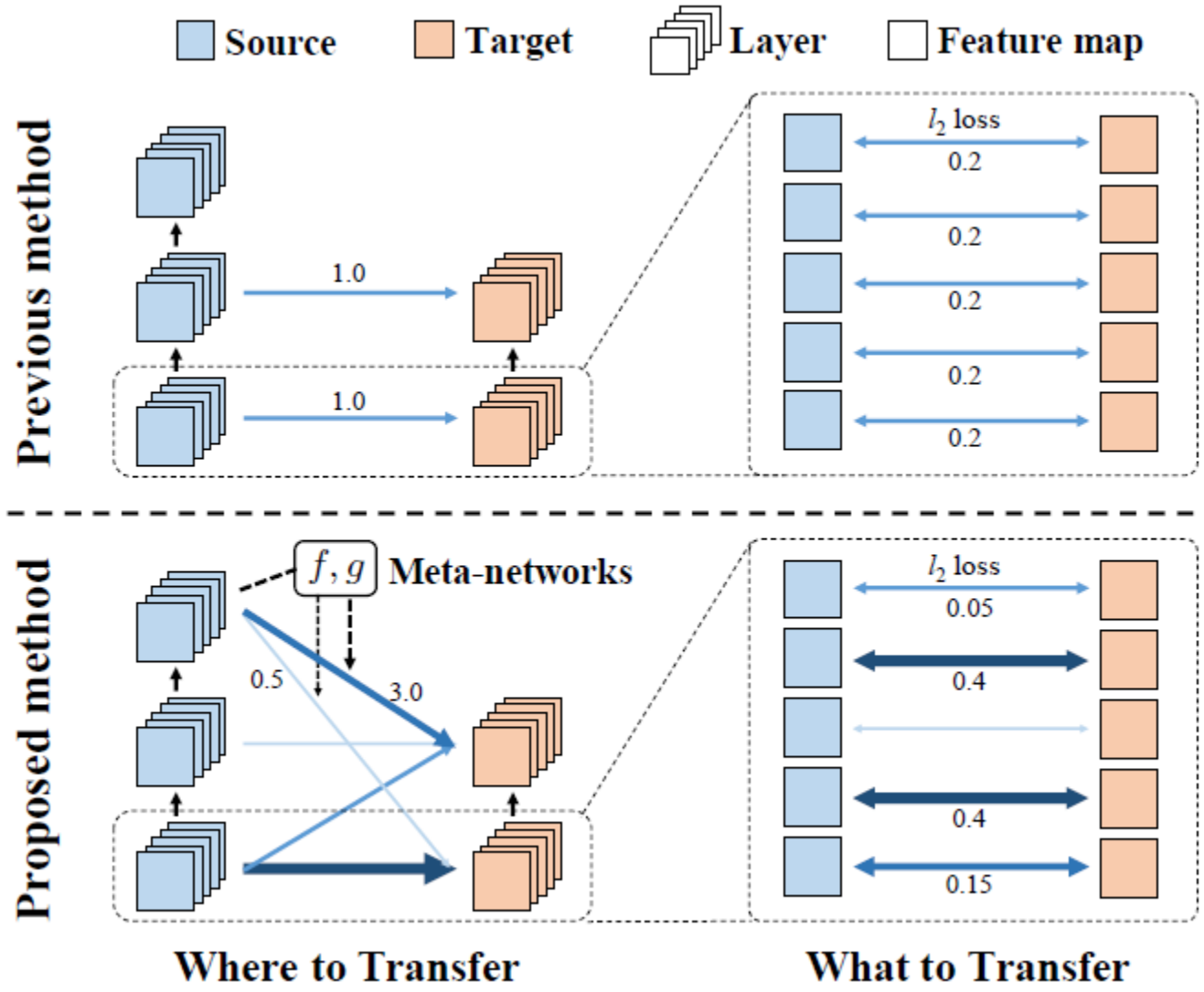
  - Some source information is more important than others, while some are irrelevant or even harmful depending on the task difference.

  - Under heterogeneous network architectures, it is not straightforward to associate a layer from the source network with one from the target network.

- ➢ Learn meta-networks that generate the weights for each feature (what) and between each pair of source and target layers (where), jointly with the target network.


Input Volume (+pad 1) (7x7x3)
x[:,:,0]
0 0 0 0 0 0 0
0 0 2 2 1 0 0
0 0 1 2 0 1 0
0 1 2 1 1 0 0
0 0 2 1 0 1 0
0 0 0 1 0 2 0
0 0 0 0 0 0 0
x[:,:,1]
0 0 0 0 0 0 0
0 1 1 1 1 0 0
0 2 1 1 0 1 0
0 1 2 0 2 2 0
0 2 0 2 0 1 0
0 2 2 0 1 1 0
0 0 0 0 0 0 0
x[:,:,2]
0 0 0 0 0 0 0
0 0 0 1 0 0 0
0 0 1 0 1 2 0
0 2 0 1 1 0 0
0 0 1 0 0 2 0
0 2 1 2 2 0 0
0 0 0 0 0 0 0
Filter W0 (3x3x3)
w0[:,:,0]
1 0 0
0 0 -1
0 -1 1
w0[:,:,1]
-1 0 0
1 1 1
1 0 1
w0[:,:,2]
1 0 1
0 0 1
-1 -1 0
Bias b0 (1x1x1)
b0[:,:,0]
1
Filter W1 (3x3x3)
w1[:,:,0]
1 1 1
1 0 -1
-1 0 -1
w1[:,:,1]
0 -1 -1
-1 0 1
1 0 1
w1[:,:,2]
-1 -1 -1
1 -1 -1
0 -1 1
Bias b1 (1x1x1)
b1[:,:,0]
0
Output Volume (3x3x2)
o[:,:,0]
3 1 -2
5 5 3
7 9 3
o[:,:,1]
0 1 -2
-6 -3 -5
-2 -4 -1

# Illustration



# Content

- Introduction
- Method
- Experiment

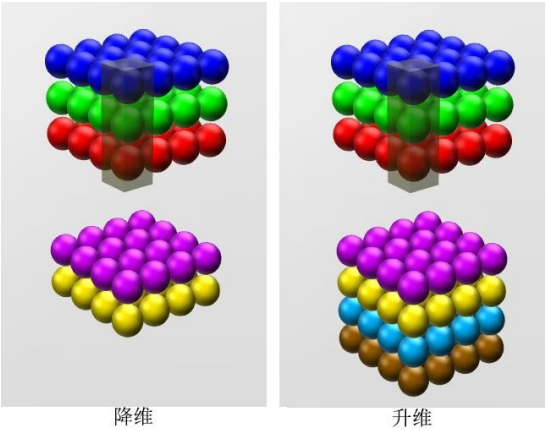
# Weighted Feature Matching

- $\{x\}$ and $\{y\}$ are images and their class labels
- Let $S^m(x)$ be intermediate feature maps of the m$^{th}$ layer of the pre-trained source network $S$
- Let $T_\theta^n(x)$ be intermediate feature maps of the n$^{th}$ layer of the target network
- The goal is to train another target network $T_\theta$ with parameter $\theta$ utilizing the knowledge of $S$

- Minimize the following $\ell_2$ objective to transfer the knowledge from $S^m(x)$ to $T_\theta^n(x)$

$$\|r_\theta(T_\theta^n(x)) - S^m(x)\|_2^2$$

where $r_\theta$ is a linear transformation parameterized by $\theta$ such as a pointwise convolution.



# What to transfer

- weighted feature matching loss

layer index

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

channel

feature map

$$\sum_c w_c^{m,n} = 1$$

$$w^{m,n} = [w_c^{m,n}] = f_\phi^{m,n}(S^m(x))$$

softmax

# Where to transfer

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

where C be a set of candidate pairs

$$\lambda^{m,n} = g^{m,n}_{\phi}(S^m(x))$$

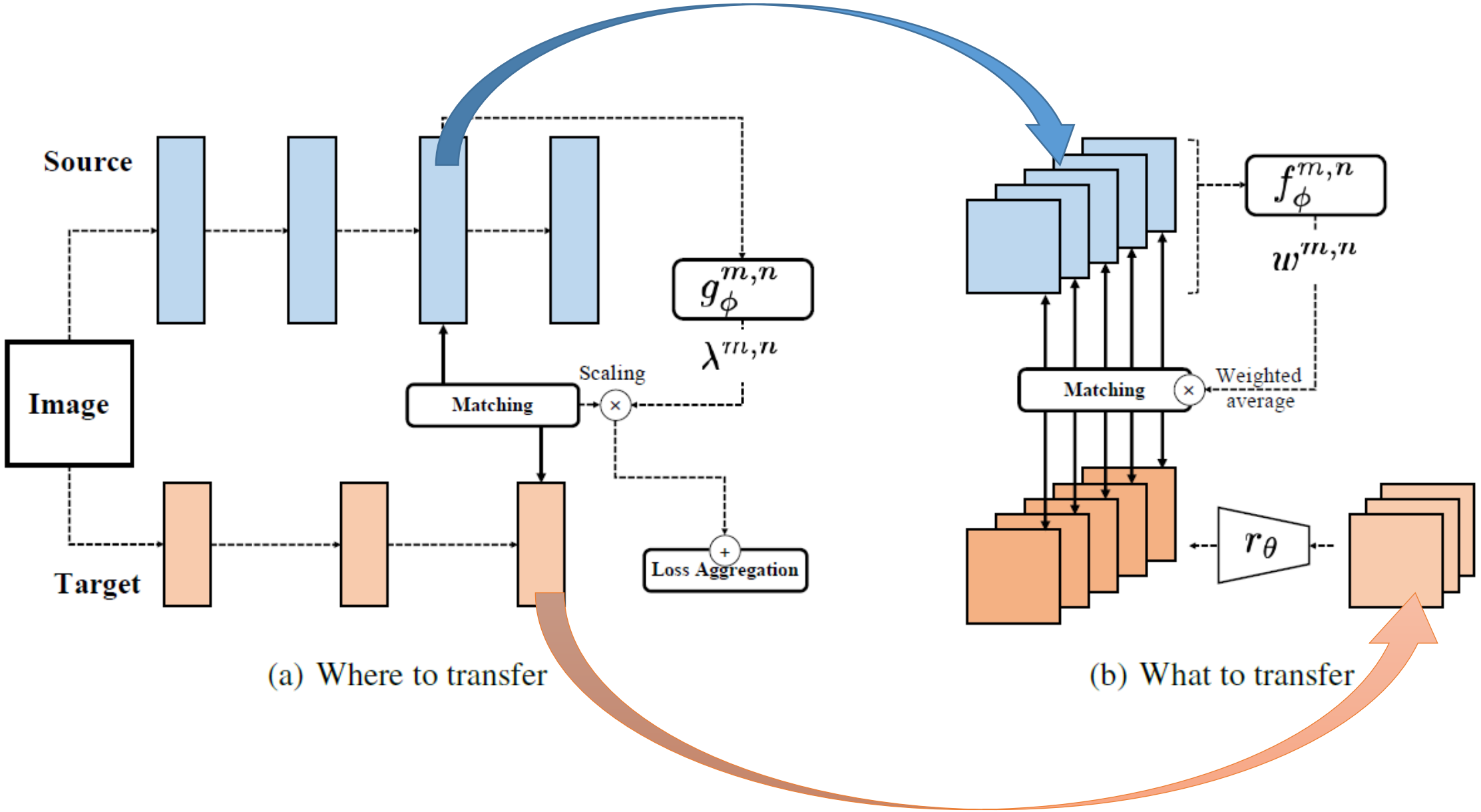
decide the amount of transfer between the $m^{th}$ and $n^{th}$ layers of source and target models

# Loss Function

$$\mathcal{L}_{\mathtt{total}}(\theta|x, y, \phi) = \mathcal{L}_{\mathtt{org}}(\theta|x, y) + \beta\mathcal{L}_{\mathtt{wfm}}(\theta|x, \phi)$$

where $L_{org}$ is the original loss (e.g., cross entropy) and $\beta$ is a hyper-parameter.

# What and Where to Transfer



# Training Meta-Networks and Target Model

$\theta$: Target model parameters
$\phi$: Meta network parameters

**Algorithm 1** Learning of $\theta$ with meta-parameters $\phi$

**Input:** Dataset $\mathcal{D}_{\texttt{train}} = \{(x_i, y_i)\}$, learning rate $\alpha$
**repeat**
  Sample a batch $\mathcal{B} \subset \mathcal{D}_{\texttt{train}}$ with $|\mathcal{B}| = B$
  Update $\theta$ to minimize $\frac{1}{B}\sum_{(x,y)\in\mathcal{B}} \mathcal{L}_{\texttt{total}}(\theta|x, y, \phi)$
  Initialize $\theta_0 \leftarrow \theta$
  **for** $t = 0$ **to** $T - 1$ **do**
    $\theta_{t+1} \leftarrow \theta_t - \alpha\nabla_\theta \frac{1}{B}\sum_{(x,y)\in\mathcal{B}} \mathcal{L}_{\texttt{wfm}}(\theta_t|x, \phi)$
  **end for**
  $\theta_{T+1} \leftarrow \theta_T - \alpha\nabla_\theta \frac{1}{B}\sum_{(x,y)\in\mathcal{B}} \mathcal{L}_{\texttt{org}}(\theta_T|x, y)$
  Update $\phi$ using $\nabla_\phi \frac{1}{B}\sum_{(x,y)\in\mathcal{B}} \mathcal{L}_{\texttt{org}}(\theta_{T+1}|x, y)$
**until** done

$\theta_T$ is learned only using the knowledge of the source model.

# Content

- Introduction
- Method
- Experiment

# Datasets and Network

| Image scale | Source | Target |
|---|---|---|
| 32*32 | TinyImageNet (32-layer ResNet) | CIFAR-10, CIFAR-100, STL-10 (9-layer VGG) |
| 224*224 | ImageNet (34-layer ResNet) | Caltech-UCSD, Bird 200, MIT Indoor Scene Recognition, Stanford 40 Actions, Stanford Dogs (18-layer ResNet) |

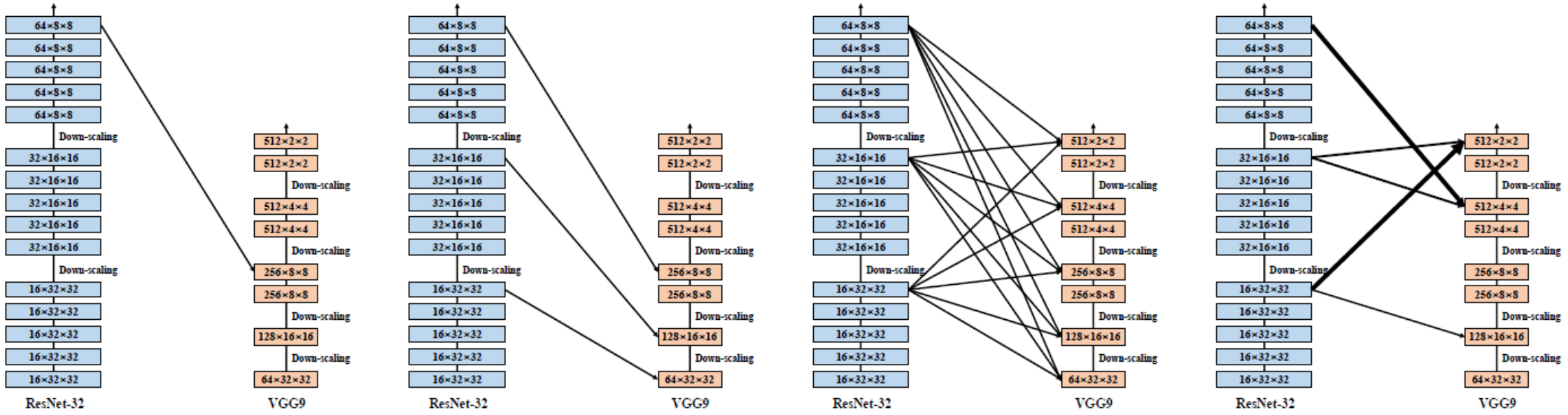
Meta-network:

- 1-layer fully-connected networks for each pair
- Input: average pooled features of the $m^{th}$ layer of the source network
- Output: $w_c^{m,n}$ and $\lambda^{m,n}$

# Compared schemes

- Learning without forgetting (LwF)
- Attention transfer (AT)
- Unweighted feature matching (FM)

matching attention maps or feature maps between source and target layers




(a) Single (b) One-to-one (c) All-to-all (d) Learned matching

*Figure 3.* (a)-(c) Matching configurations $\mathcal{C}$ between ResNet32 (left) and VGG9 (right). (d) The amount $\lambda^{m,n}$ of transfer between layers after learning. Line widths indicates the transfer amount. We omit the lines when $\lambda^{m,n}$ is less than 0.1.

# Evaluation on Various Target Tasks

*Table 1.* Classification accuracy (%) of transfer learning from TinyImageNet ($32 \times 32$) or ImageNet ($224 \times 224$) to CIFAR-100, STL-10, Caltech-UCSD Bird 200 (CUB200), MIT Indoor Scene Recognition (MIT67), Stanford 40 Actions (Stanford40) and Stanford Dogs datasets. For TinyImageNet, ResNet32 and VGG9 are used as a source and target model, respectively, and ResNet34 and ResNet18 are used for ImageNet.

| 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 | 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 (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+AT (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 (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 (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 |

# Experiments on Limited-Data Regimes

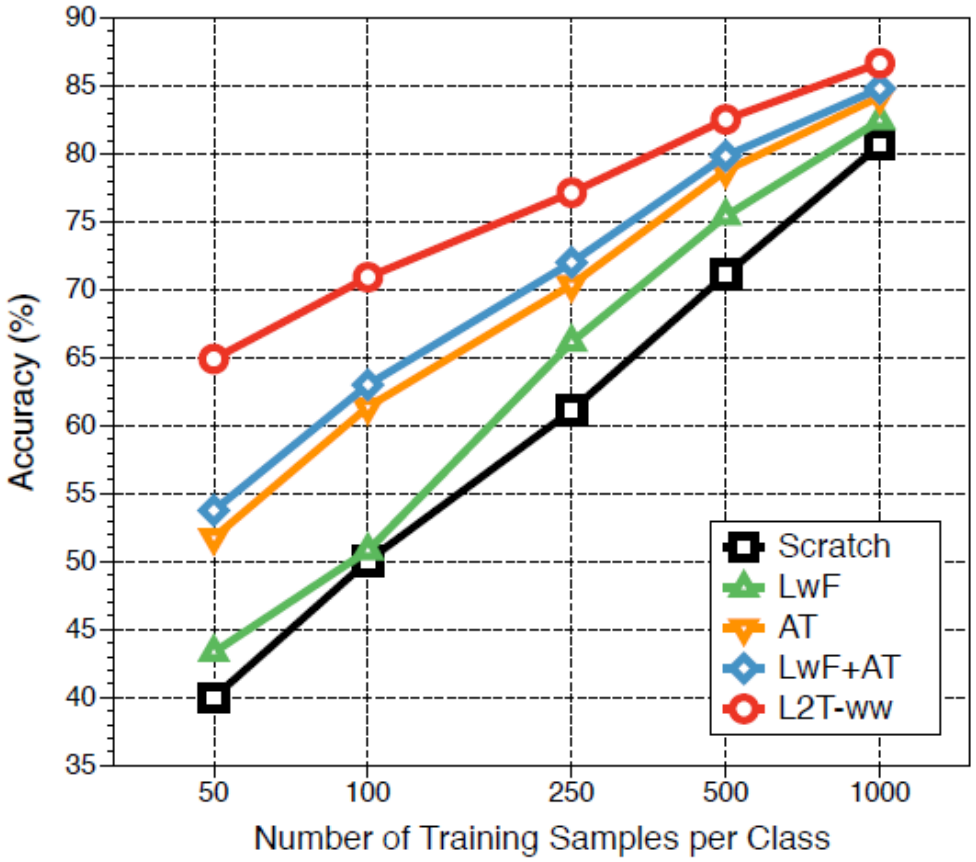



*Figure 5.* Transfer from TinyImageNet to CIFAR-10 with varying numbers of training samples per class in CIFAR-10. $x$-axis is plotted in logarithmic scale.

# Experiments on Multi-Source Transfer

*Table 2.* Classification accuracy (%) of VGG9 on STL-10 transferred from multiple source models. The first source model is ResNet32 trained on TinyImageNet. The additional source model is one of three: ResNet20 trained on TinyImageNet, another ResNet32 trained on TinyImageNet, and ResNet32 trained on CIFAR-10. We report the performance of the target model transferred from a single source model and two source models.

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

# Visualization

pixels are more or less activated in the saliency map of L2T-w compared to FM


*Figure 6.* More (the second column) and less (the third column) activated pixels in the saliency maps of L2T-w compared to unweighted feature matching (FM) on images of (a) CUB200 and (b) Stanford Dogs datasets. When computing saliency maps, we use normalized gradients. One can observe that the higher activated pixels induced by L2T-w tend to correspond to where task-specific objects are, while less activated location spread over entire location.