Transfer learning and Domain adaptation

Vision & Language
Transfer from ImageNet (source)

**Transfer as generic features**
Brut Deep features (learned from ImageNet)
(== a learned embedding from Image to vector representation)

Retrieval

**Transfer learning (from source to target)**
Frozen features + SVM => solution to small datasets
Frozen features + Deep
Fine tuning not easy in that case (small datasets)
Transfer from source (=ImageNet task) to target task

**Source:** ImageNet (dataset + 100 classes) => AlexNet trained

**Target:** new dataset Cal-101 and new classification task with 101 classes => Chopped

AlexNet (layer i) + SVM trained on

**Architecture of Krizhevsky et al.**

- 8 layers total
- Trained on Imagenet dataset [Deng et al. CVPR'09]
- 18.2% top-5 error
- Our reimplementations: 18.1% top-5 error

**Tapping off Features at each Layer**

Plug features from each layer into linear SVM or soft-max

<table>
<thead>
<tr>
<th>SVM (layer)</th>
<th>Cal-101 (30/class)</th>
<th>Cal-256 (60/class)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM (1)</td>
<td>44.8 ± 0.7</td>
<td>24.6 ± 0.4</td>
</tr>
<tr>
<td>SVM (2)</td>
<td>66.2 ± 0.5</td>
<td>39.6 ± 0.3</td>
</tr>
<tr>
<td>SVM (3)</td>
<td>72.3 ± 0.4</td>
<td>46.0 ± 0.3</td>
</tr>
<tr>
<td>SVM (4)</td>
<td>76.6 ± 0.4</td>
<td>51.3 ± 0.1</td>
</tr>
<tr>
<td>SVM (5)</td>
<td><strong>86.2 ± 0.8</strong></td>
<td>65.6 ± 0.3</td>
</tr>
<tr>
<td>SVM (7)</td>
<td><strong>85.5 ± 0.4</strong></td>
<td><strong>71.7 ± 0.2</strong></td>
</tr>
<tr>
<td>Softmax (5)</td>
<td>82.9 ± 0.4</td>
<td>65.7 ± 0.5</td>
</tr>
<tr>
<td>Softmax (7)</td>
<td><strong>85.4 ± 0.4</strong></td>
<td><strong>72.6 ± 0.1</strong></td>
</tr>
</tbody>
</table>

=> Results better than SoA CV methods on Cal-101!
Transfer: fine-tuning of a deep model on target task

Train a deep (AlexNet) on source (ImageNet)
Keep the deep params. for target and complete with a small deep on top (fully trained on target task)
Fine-tune the whole model on target data
  Challenge: only limited target data, careful about overfitting
  Solution: Freeze the gradient’s update for AlexNet part

Source

Pretrained Model

Copy

Target

Fixed Weights

New Classifier

Gradient Blocked
Transfer: fine-tuning of a deep model on target task

Train a deep (AlexNet) on source (ImageNet)
Keep the deep params. for target and complete with a small deep on top (fully trained on target task)
Fine-tune the whole model on target data
  Challenge: only limited target data, careful about overfitting
  Solution: Freeze the gradient’s update for AlexNet part
Other solution: use smaller gradient’s update for AlexNet part
Transfer: which parts of the deep?

Which layer(s) can be transferred (copied)?

- Speech: usually copy the last few layers
- Image: usually copy the first few layers
Transfer: which supervision?

• Task description
  • Source data: \((x^s, y^s)\) \rightarrow A large amount
  • Target data: \((x^t, y^t)\) \rightarrow (Very) little

Rq: Few/One-shot learning: only a few/one examples in target domain

Many different contexts:
In vision: from large dataset (ImageNet) to small datasets (VOC2007)
In speech: (supervised) speaker adaption
  • Source data: audio data and transcriptions from many speakers
  • Target data: audio data and its transcriptions of specific user
More on transfer framework

<table>
<thead>
<tr>
<th>Target Data</th>
<th>Source Data (ImageNet)</th>
</tr>
</thead>
<tbody>
<tr>
<td>labelled</td>
<td>Frozen or fine-tuning</td>
</tr>
<tr>
<td></td>
<td>Few</td>
</tr>
<tr>
<td></td>
<td>One</td>
</tr>
<tr>
<td></td>
<td>Zero</td>
</tr>
</tbody>
</table>

Main purposes:
Similar visual domain?
Same tasks (ie class)?
Similar domain: ImageNet task => Dog/Cat task

Target: Dog/Cat Classifier

Data *not directly related to* the task considered

ImageNet: Similar domain, different task (1000 classes but NOT Dog and Cat classes)
General Framework for Transfer Learning

Target: Dog/Cat Classifier

Data *not directly related to* the task considered

Similar domain, completely different tasks

Different domains, same task
<table>
<thead>
<tr>
<th>Target Data</th>
<th>Source Data (not directly related to the task)</th>
</tr>
</thead>
<tbody>
<tr>
<td>labelled</td>
<td>labelled</td>
</tr>
<tr>
<td>Fine-tuning</td>
<td>Multitask Learning</td>
</tr>
<tr>
<td>Domain-adversarial training</td>
<td></td>
</tr>
<tr>
<td>Zero-shot learning</td>
<td></td>
</tr>
<tr>
<td>Not considered here</td>
<td>Self-supervised</td>
</tr>
<tr>
<td></td>
<td>Self-taught learning</td>
</tr>
<tr>
<td></td>
<td>Self-taught Clustering</td>
</tr>
</tbody>
</table>
### General Framework for Transfer Learning

<table>
<thead>
<tr>
<th>Source Data (not directly related to the task)</th>
<th>Target Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>labelled</td>
<td>labelled</td>
</tr>
<tr>
<td><strong>Fine-tuning</strong></td>
<td><strong>Multitask Learning</strong></td>
</tr>
<tr>
<td>unlabelled</td>
<td>unlabelled</td>
</tr>
</tbody>
</table>

Not considered here
Multitask Learning

- The multi-layer structure makes NN suitable for multitask learning

Task A

Task B

Input feature

Input feature for task A

Input feature for task B
## Transfer Learning - Overview

<table>
<thead>
<tr>
<th>Target Data</th>
<th>Source Data (not directly related to the task)</th>
<th>Source Data (not directly related to the task)</th>
</tr>
</thead>
<tbody>
<tr>
<td>labelled</td>
<td>Fine-tuning</td>
<td>Not considered here</td>
</tr>
<tr>
<td>labelled</td>
<td><em>Multitask Learning</em></td>
<td>Not considered here</td>
</tr>
<tr>
<td>labelled</td>
<td><em>Domain adaptation-adversarial training</em></td>
<td>Not considered here</td>
</tr>
<tr>
<td>labelled</td>
<td></td>
<td></td>
</tr>
<tr>
<td>labelled</td>
<td></td>
<td></td>
</tr>
<tr>
<td>labelled</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Unsupervised Domain Adaptation (UDA)

Source data: $(x^s, y^s)$ → Training data
Target data: $(x^t)$ → Same task, domain mismatch

Source with labels
Target without labels

Final test on target domain!
Main principle: diminish the domain shift in the learned features, encourage domain confusion
UDA strategy: align both domains
UDA strategy: 1/ domain-adversarial training

Add to the feature generator (G) a domain classifier (discriminant D) for which labels are available!

Learn G and D:
- G tries to align domains
- D tries to identify domains

Rq: Similar to GAN (coming soon)
UDA strategy: 1/ domain-adversarial training
   2/ classification task (same for source and target here)

Maximize label classification accuracy + minimize domain classification accuracy

Feature extractor

Not only cheat the domain classifier, but satisfying label classifier at the same time

Label predictor

Maximize label classification accuracy

Domain classifier

Maximize domain classification accuracy
UDA strategy: joint learning

Domain classifier fails in the end
It should struggle ……

Yaroslav Ganin, Victor Lempitsky, Unsupervised Domain Adaptation by Backpropagation, ICML, 2015

Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, Domain-Adversarial Training of Neural Networks, JMLR, 2016
## Domain-adversarial training

<table>
<thead>
<tr>
<th></th>
<th>MNIST</th>
<th>SYN Numbers</th>
<th>SVHN</th>
<th>SYN Signs</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Source</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Target</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>METHOD</strong></td>
<td><strong>SOURCE</strong></td>
<td><strong>TARGET</strong></td>
<td><strong>SOURCE</strong></td>
<td><strong>TARGET</strong></td>
</tr>
<tr>
<td><strong>SOURCE ONLY</strong></td>
<td>MNIST</td>
<td>MNIST-M</td>
<td>SYN Numbers</td>
<td>SVHN</td>
</tr>
<tr>
<td></td>
<td>.5749</td>
<td>.8665</td>
<td>.5919</td>
<td>.7400</td>
</tr>
<tr>
<td><strong>SA (FERNANDO ET AL., 2013)</strong></td>
<td>.6078 (7.9%)</td>
<td>.8672 (1.3%)</td>
<td>.6157 (5.9%)</td>
<td>.7635 (9.1%)</td>
</tr>
<tr>
<td><strong>PROPOSED APPROACH</strong></td>
<td><strong>.8149 (57.9%)</strong></td>
<td><strong>.9048 (66.1%)</strong></td>
<td><strong>.7107 (29.3%)</strong></td>
<td><strong>.8866 (56.7%)</strong></td>
</tr>
<tr>
<td><strong>TRAIN ON TARGET</strong></td>
<td>.9891</td>
<td>.9244</td>
<td>.9951</td>
<td>.9987</td>
</tr>
</tbody>
</table>

Yaroslav Ganin, Victor Lempitsky, Unsupervised Domain Adaptation by Backpropagation, ICML, 2015

Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, Domain-Adversarial Training of Neural Networks, JMLR, 2016
Domain adaptation

Main principle: diminish the domain shift in the learned features, encourage domain confusion

Another example: Adversarial Discriminative Domain Adaptation [Tzeng et al. 2017]
Domain adaptation

Other architecture
Domain adaptation

Other architecture: Image translation for Domain adaptation [Murez 2017]
Domain adaptation

Other architecture: Image translation for Domain adaptation [Murez 2017]
Use-Case: Domain adaptation for Autonomous driving
Context: Neural network-based autonomous driving system framework

**Sensors**
- Camera
- RADAR
- LiDAR
- IMU
- GPS

**Inputs**
- Local history
- Point clouds
- RGB Video
- Object detections
- Semantic segmentations
- Depth maps
- Bird-eye-view

**Learning**
- Imitation learning with a dataset
- Reinforcement learning with a simulator

**Outputs**
- Vehicle controls
- Future trajectory
Challenges for perception

Multi-sensor perception
• Sensor fusion; Camera, radar and Lidar

3D dynamic understanding
• 3D object detection; Motion forecast; Intention prediction

Frugal learning
• Training with limited data or supervision; Domain adaptation

Reliability
• Robustness; Uncertainty estimation; Failure prediction

Explainability
• Decision interpretation; Post-hoc or by-design
Domain gap

Different, though *related* input data distributions

**Source** domain → **Target** domain

- Different weather, light, location, sensor’s spec/setup
Domain gap

Different, though *related* input data distributions

Source domain $\rightarrow$ Target domain

- Different weather, light, location, sensor’s spec/setup
Domain gap

Different, though *related* input data distributions

Source domain → Target domain

- Different weather, light, location, sensor’s spec/setup
Domain gap

Different, though *related* input data distributions

Source domain → Target domain

- Different weather, light, location, sensor’s spec/setup
Domain gap

Different, though *related* input data distributions

Source domain → Target domain

- Synthetic vs. real
Domain gap for **VISUAL SEGMENTATION**

Different, though *related* input data distributions

*Source domain → Target domain*

- Synthetic vs. real
Domain gap

Different, though *related* input data distributions

Source domain $\rightarrow$ Target domain

- Synthetic vs. real
Unsupervised Domain Adaptation (UDA)

Labelled source domain data

Unlabelled target domain data
Unsupervised Domain Adaptation (UDA)

**TRAIN**

- Source labelled data

**TEST**

- Source
- Target

Learned segmentation model
Unsupervised Domain Adaptation (UDA)

**TRAIN**

Source labelled data

learned segmentation model

**TEST**

Source

Target
Unsupervised Domain Adaptation (UDA)

**TRAIN**

Source labelled data

**TEST**

Source

Target

learned segmentation model
Unsupervised Domain Adaptation (UDA)

<table>
<thead>
<tr>
<th>TRAIN</th>
<th>TEST</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source labelled data</td>
<td>Source</td>
</tr>
<tr>
<td>learned segmentation model</td>
<td>Target</td>
</tr>
</tbody>
</table>
Expected results with UDA training

<table>
<thead>
<tr>
<th>TRAIN</th>
<th>TEST</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source labelled data</td>
<td>Source</td>
</tr>
<tr>
<td></td>
<td>Target</td>
</tr>
</tbody>
</table>

Source segmentation model
Unsupervised Domain Adaptation (UDA)

Top layer alignment on $P_x$ or $I_x$

Source

Target

$x_s$ $x_t$

$y_s$

Soft-segmentation map

$L_{seg}(x_s, y_s)$

$L_D(I_{x_s}, 1) + L_D(I_{x_t}, 0)$

$L_D(I_{x_t}, 1)$

Entropy minimization with Adversarial training
Qualitative results

input image

without UDA

with UDA
UDA Results (with Adversarial Entropy)
Extension: Zero shot + Domain adaptation

input image with UDA

GT

without UDA

## Transfer Learning - Overview

<table>
<thead>
<tr>
<th>Source Data (not directly related to the task)</th>
<th>Target Data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>labelled</td>
</tr>
<tr>
<td></td>
<td>Fine-tuning</td>
</tr>
<tr>
<td></td>
<td>Multitask Learning</td>
</tr>
<tr>
<td></td>
<td>unlabelled</td>
</tr>
<tr>
<td></td>
<td>Not considered here</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Zero-shot Learning

- Source data: \((x^s, y^s)\) \(\rightarrow\) Training data
- Target data: \((\emptyset)\) usually same domain

Different tasks

Training time:
\[
x^s: \\
y^s: \text{cat} \quad \text{dog} \quad \ldots
\]

Test time \(x^t:\)
\[
\Rightarrow \text{Fish class!}
\]
Zero-shot Learning

- Representing each class by its attributes

**Training**

<table>
<thead>
<tr>
<th>1</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>furry</td>
<td>4 legs</td>
<td>tail</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>furry</td>
<td>4 legs</td>
<td>tail</td>
</tr>
</tbody>
</table>

Database attributes

<table>
<thead>
<tr>
<th>class</th>
<th>furry</th>
<th>4 legs</th>
<th>tail</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dog</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>Fish</td>
<td>X</td>
<td>X</td>
<td>O</td>
</tr>
<tr>
<td>Chimp</td>
<td>O</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Sufficient attributes for one to one mapping
Zero-shot Learning

- Representing each class by its attributes

**Testing**

Find the class with the most similar attributes

```
| attributes | furry | 4 legs | tail | ...
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Dog</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>Fish</td>
<td>X</td>
<td>X</td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>Chimp</td>
<td>O</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

sufficient attributes for one to one mapping
Zero-shot Learning

- Attribute embedding + class (word name) embedding

What if we don't have attribute database
Zero-shot Learning

- Attribute embedding

\[ f(*) \text{ and } g(*) \text{ can be NN.} \]

Training target:

\[ f(x^n) \text{ and } g(y^n) \text{ as close as possible} \]

\[ y^1 \text{ (attribute of chimp)} \]

\[ y^2 \text{ (attribute of dog)} \]

\[ y^3 \text{ (attribute of Fish)} \]

\[ y^i \text{ are linked together by a class relationship (e.g. class name embedding as W2v)} \]
More on Vision-Language Embedding Space

- chimp
- dog
- fish

Embedding Space
CLIP: Vision + Language Models (VLM)

[Learning transferable visual models from natural language supervision. Radford/Sutskever ICML, 2021]

Embedding Space

A dog runs in the garden

Chimp sits against the wall

a pretty orange fish swims with other fishes
CLIP: Vision + Language Models (VLM)

Dual architecture: Text encoder + Image encoder
CLIP: Vision + Language Models (VLM)

**Learning strategy**
Training set: \( A = \{ (I_n, T_n) \}_{n} \) of image/caption pairs (coherent!)
Massive Text+Image = 400M pairs to train the model (from the Internet)
Contrastive loss for training: positive pair vs negative one or set

A dog runs in the garden

Massive Text+Image = 400M pairs to train the model (from the Internet)
Contrastive loss for training: positive pair vs negative one or set
**Learning strategy**

Training set: \( A = \{(I_n, T_n)\}_n \) of image/caption pairs (coherent!)

Massive Text+Image = **400M pairs** to train the model (from the Internet)

Contrastive loss for training: positive pair vs negative one or set

---

<table>
<thead>
<tr>
<th>A dog runs in the garden</th>
<th><strong>Text Encoder</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Image Encoder</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( I_1 )</th>
<th>( I_2 )</th>
<th>( I_3 )</th>
<th>( \vdots )</th>
<th>( I_N )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( I_1 \cdot T_1 )</td>
<td>( I_1 \cdot T_2 )</td>
<td>( I_1 \cdot T_3 )</td>
<td>( \vdots )</td>
<td>( I_1 \cdot T_N )</td>
</tr>
<tr>
<td>( I_2 \cdot T_1 )</td>
<td>( I_2 \cdot T_2 )</td>
<td>( I_2 \cdot T_3 )</td>
<td>( \vdots )</td>
<td>( I_2 \cdot T_N )</td>
</tr>
<tr>
<td>( I_3 \cdot T_1 )</td>
<td>( I_3 \cdot T_2 )</td>
<td>( I_3 \cdot T_3 )</td>
<td>( \vdots )</td>
<td>( I_3 \cdot T_N )</td>
</tr>
<tr>
<td>( \vdots )</td>
<td>( \vdots )</td>
<td>( \vdots )</td>
<td>( \vdots )</td>
<td>( \vdots )</td>
</tr>
<tr>
<td>( I_N \cdot T_1 )</td>
<td>( I_N \cdot T_2 )</td>
<td>( I_N \cdot T_3 )</td>
<td>( \vdots )</td>
<td>( I_N \cdot T_N )</td>
</tr>
</tbody>
</table>

CLIP: Vision + Language Models (VLM)
CLIP: Vision + Language Models (VLM)

**Learning strategy**
Training set: $A = \{(I_n, T_n)\}_n$ of image/caption pairs (coherent!)
Massive Text+Image = 400M pairs to train the model (from the Internet)
Contrastive loss for training: positive pair vs negative pair or more

(contrastive) Triplet loss: A variant of the standard margin based loss (SVM)
- Triplet $(y, z, z')$
- Anchor: $y$ (E.g image representation)
- Positive: $z$ (E.g associated caption representation)
- Negative: $z'$ (E.g contrastive caption representation)
- Margin parameter $\alpha$

$$\text{loss}(y, z, z') = \max\{0, \alpha - <y, z> + <y, z'>\}$$
Learning strategy: triplet loss

\[
\text{loss}(y, z, z') = \max \{ 0, \alpha + d(y, z) - d(y, z') \}
\]
Learning strategy: triplet loss

**Hard negative** margin-based loss:

*Loss for a batch* $\mathcal{B} = \{(I_n, T_n)\}_{n \in B}$ of image/sentence pairs:

$$
\mathcal{L}(\Theta; \mathcal{B}) = \frac{1}{|\mathcal{B}|} \sum_{n \in B} \left( \max_{m \in C_n \cap \mathcal{B}} \text{loss} (x_n, v_n, v_m) + \max_{m \in D_n \cap \mathcal{B}} \text{loss} (v_n, x_n, x_m) \right)
$$

With $C_n$ (resp. $D_n$) set of indices of caption (resp. image) unrelated to $n$-th element
Learning strategy: hard negative triplet loss

**Mining hard negative contrastive example:**

\[
\mathcal{L}(\theta; B) = \frac{1}{|B|} \sum_{n \in B} \left( \max_{m \in C_n \cap B} \text{loss} (x_n, v_n, v_m) + \max_{m \in D_n \cap B} \text{loss} (v_n, x_n, x_m) \right)
\]
Learning strategy: hard negative triplet loss

Mining hard negative contrastive example:

\[
\mathcal{L}(\Theta; \mathcal{B}) = \frac{1}{|\mathcal{B}|} \sum_{n \in \mathcal{B}} \left( \max_{m \in C_n \cap \mathcal{B}} \text{loss} (x_n, v_n, v_m) \right) + \max_{m \in D_n \cap \mathcal{B}} \text{loss} (v_n, x_n, x_m)
\]
CLIP: Vision + Language Models (VLM)

Massive Text+Image = \textbf{400M pairs} to train the model (from the Internet)

Contrastive loss for training

\[
\mathcal{L}_{\text{InfoNCE}_{\text{CLIP}}} = - \sum_i \log \left( \frac{\exp\left( \frac{\text{sim}(I_i, T_i)}{\tau} \right)}{\sum_{k=1}^{N} \exp\left( \frac{\text{sim}(I_i, T_k)}{\tau} \right)} \right)
\]
CLIP: Vision + Language Models (VLM)

Pre-trained encoders = **dual encoders** *(Text/Image)*
used for Zero-shot classifier, and other downstream tasks
A lot of variants ....

<table>
<thead>
<tr>
<th>Title query</th>
<th>Ingredient query</th>
<th>Instruction query</th>
<th>Top 5 retrieved images</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mint Chocolate Chip Frosting.</td>
<td>1 cup Unsalted Butter, 2 Tablespoons Heavy Cream, 2 drops Green Food Coloring, ...</td>
<td>Add sugar, cream, peppermint, and food coloring... ... scoop the frosting and place on top of your cupcakes Source: Chocolate Cupcakes with Mint Chocolate Chip ...</td>
<td>![Image 1](Image 1) ![Image 2](Image 2) ![Image 3](Image 3) ![Image 4](Image 4) ![Image 5](Image 5)</td>
</tr>
<tr>
<td>Honey-Grilled Chicken.</td>
<td>1 broiler-fryer chicken, halved, 34 cup butter, melted, 14 cup honey...</td>
<td>Place the halved chicken in a large, shallow container... ... Combine the remaining ingredients, stirring sauce well Grill chicken, skin side up...</td>
<td>![Image 6](Image 6) ![Image 7](Image 7) ![Image 8](Image 8) ![Image 9](Image 9) ![Image 10](Image 10)</td>
</tr>
<tr>
<td>The Best Kale Ever.</td>
<td>1/2 cup Kale, 1 teaspoon Olive Oil, 1/4 teaspoons Red Pepper Flakes...</td>
<td>Wash and cut kale off the stems... Heat olive oil on medium heat and add garlic Add in kale and red pepper flakes...</td>
<td>![Image 11](Image 11) ![Image 12](Image 12) ![Image 13](Image 13) ![Image 14](Image 14) ![Image 15](Image 15)</td>
</tr>
</tbody>
</table>