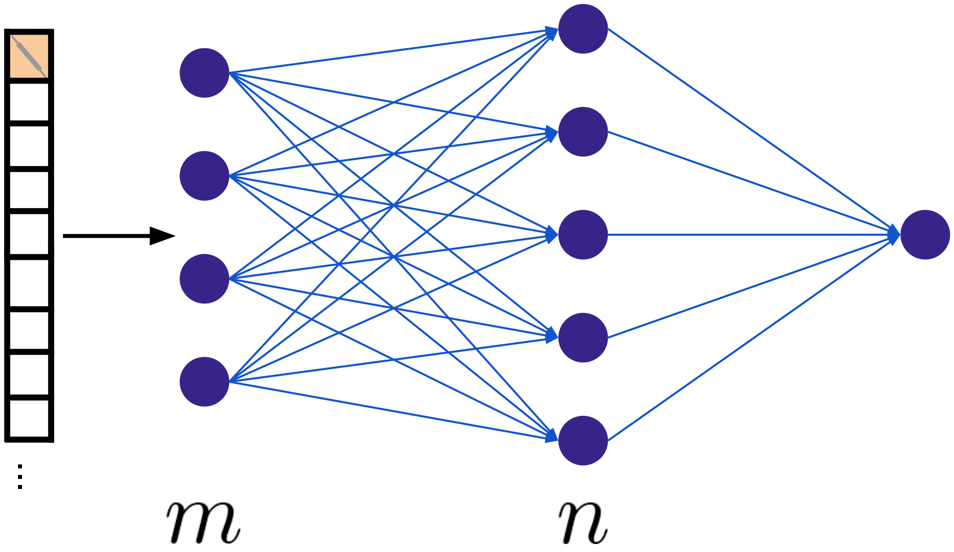
Transfer Learning

BOSTON UNIVERSITY
MACHINE INTELLIGENCE COMMUNITY

Duy Nguyen, Justin Chen
Oct. 31, 2017

CNN: Fully-connected Layer



$$\begin{bmatrix} \theta^1_{11} & \theta^1_{12} & \theta^1_{13} & \dots & \theta^1_{1m} \\ \theta^1_{21} & \theta^1_{22} & \theta^1_{23} & \dots & \theta^1_{2m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \theta^1_{n1} & \theta^1_{n2} & \theta^1_{n3} & \dots & \theta^1_{nm} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix}$$

$$\begin{bmatrix} \theta^2_{11} & \theta^2_{12} & \theta^2_{13} & \dots & \theta^2_{1n} \end{bmatrix} \sigma\left(\begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} \right)$$

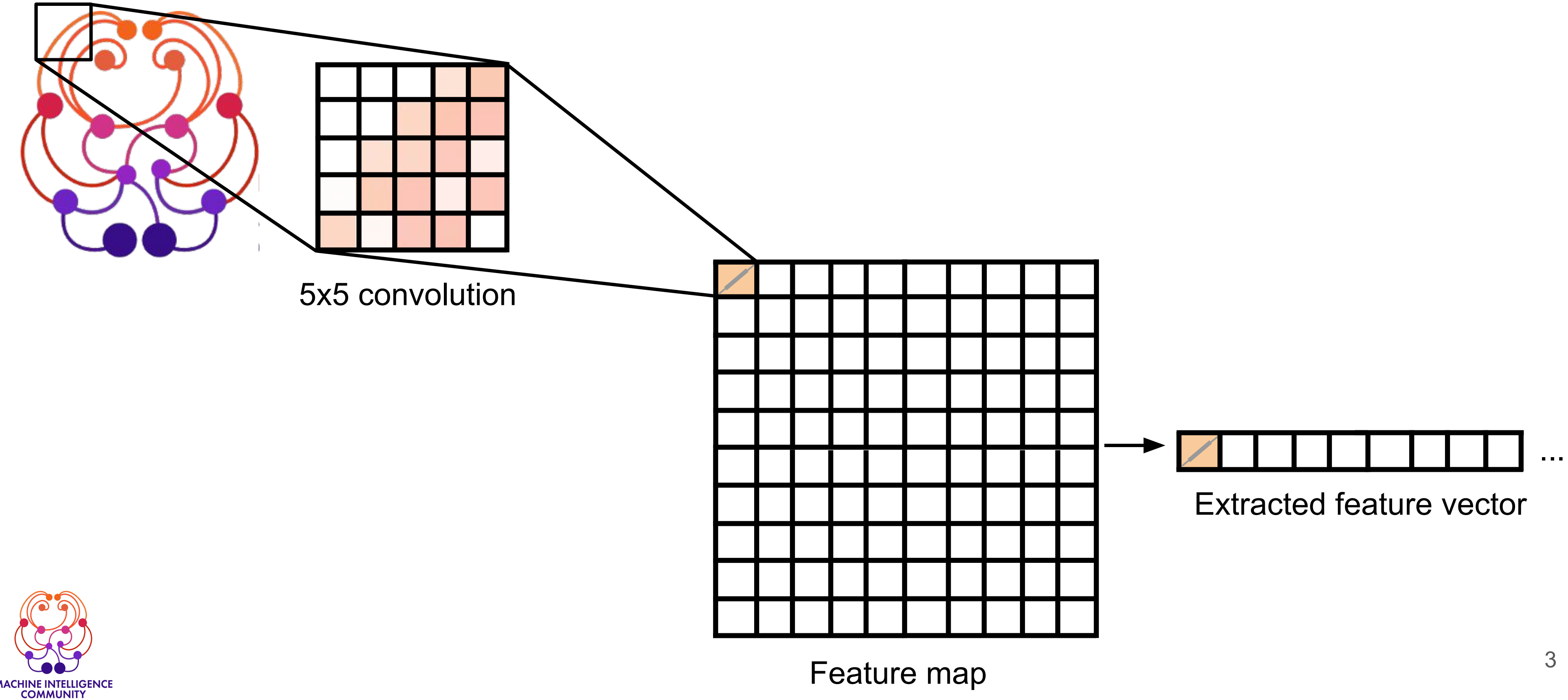
$$= \begin{bmatrix} h_1 \end{bmatrix}$$

Ignoring bias values for simplification

In this example, we're doing regression

MACHINE INTELLIGENCE COMMUNITY

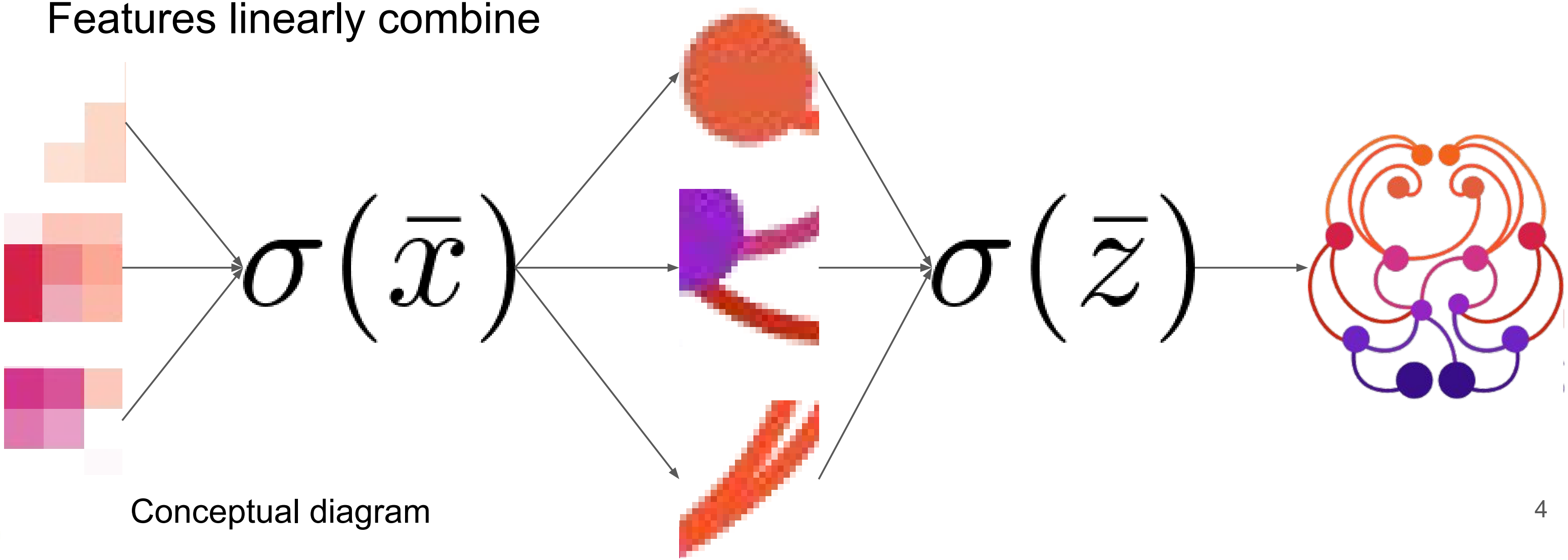
CNN: Automatic Feature Extraction



MACHINE INTELLIGENCE COMMUNITY

Feature Hierarchy

- Learned features become progressively more complex throughout the network
- Earlier layers contain more primitive features
- Features linearly combine



Conceptual diagram

Primitive features

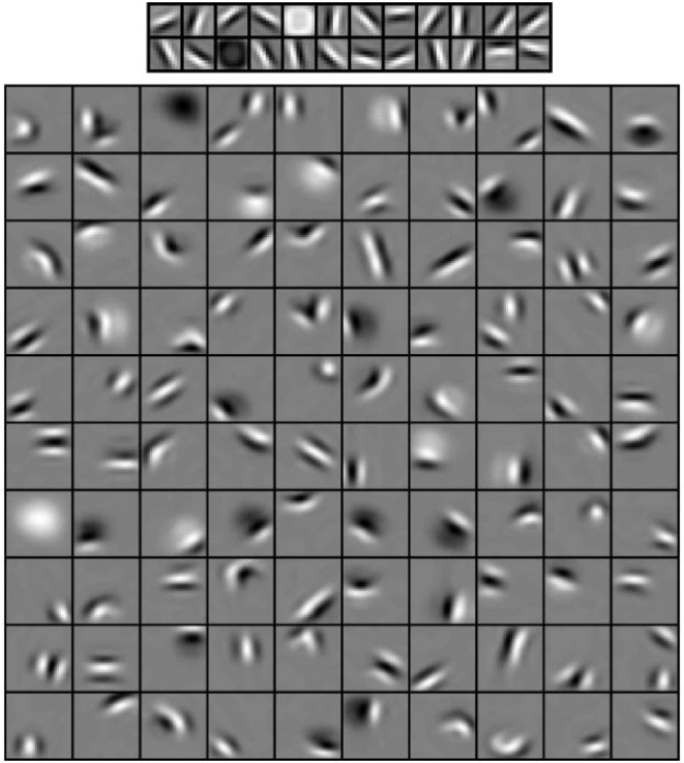


Image from Lee et al. 2011

Complex features

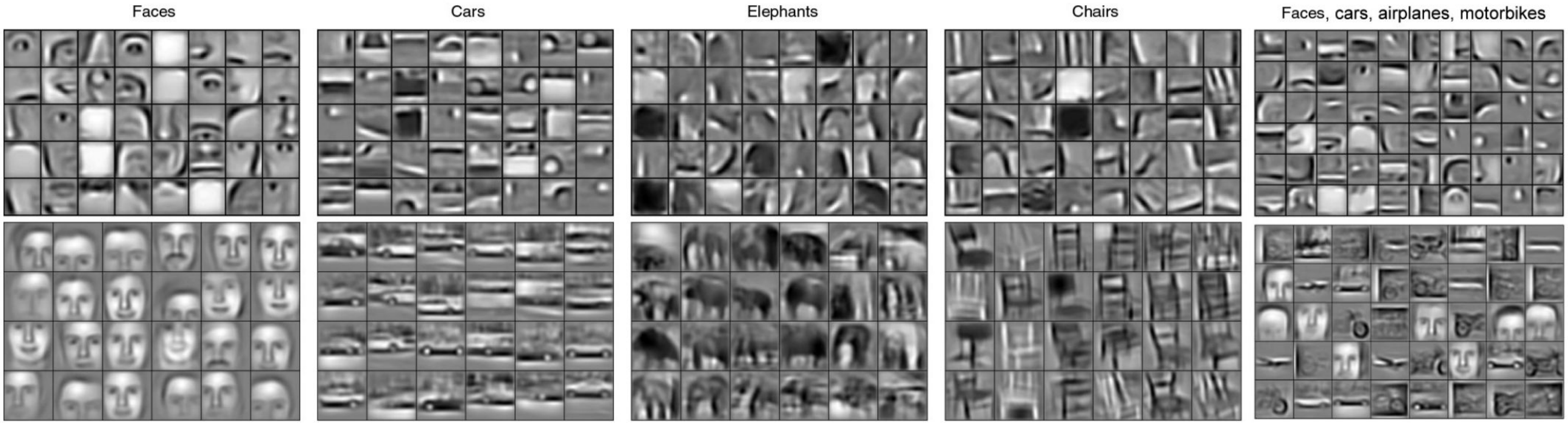
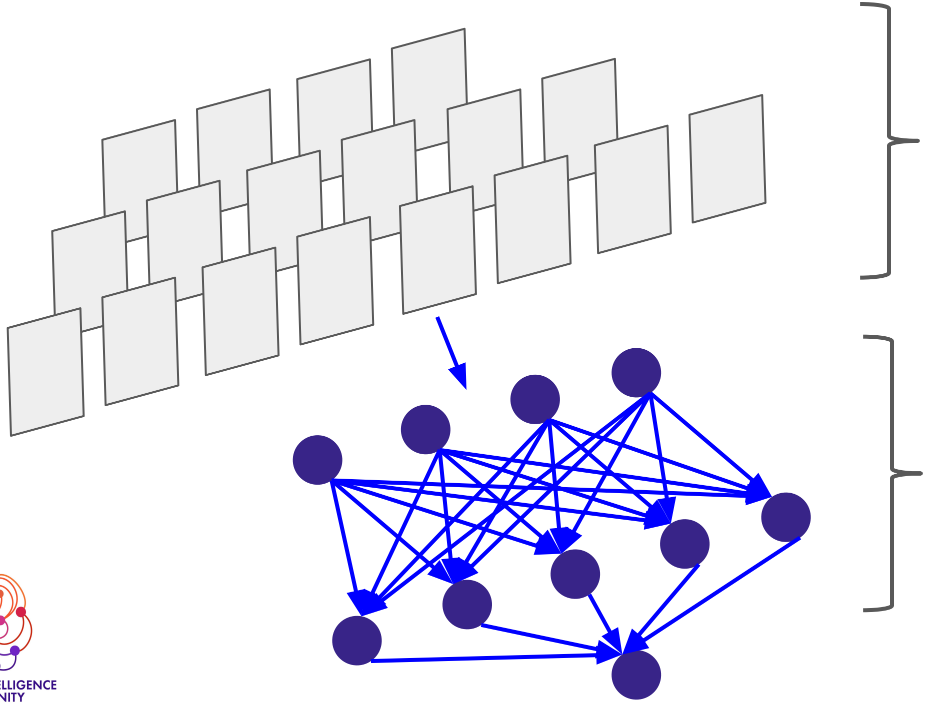


Image from Lee et al. 2011

How to reuse learned models?

- Would be useful to be able to reuse learned parameters and learned features



- Trained feature extractor contains features common among domain
- Reuse this part

- Trained fully-connected network contains features specific to dataset
- Can potentially reuse weights in latter layers as well

MACHINE INTELLIGENCE COMMUNITY

Deep learning so far ...

- Convolutional neural networks allows us to capture compositional data and convert it into a hierarchical representation.
  - This approach have prove quite effective in classification task, especially in computer vision.
- While CNN is effective in computer vision task, it is also requires a lot of resources.
- CNN, Neural Networks, Backpropagation are not new. All of these techniques have been around since the 1980s. But why it’s popular now?

MACHINE INTELLIGENCE COMMUNITY

ImageNet

- Both a collection of 14+ millions images that are categorised **by human** into roughly 20 thousands categories. This dataset also fuels the ImageNet challenge.

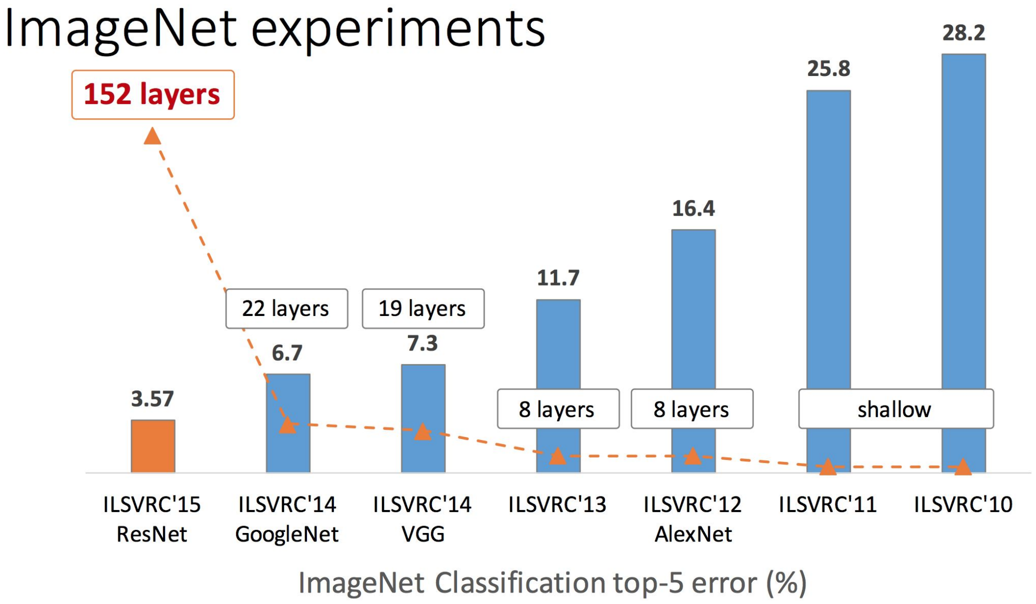
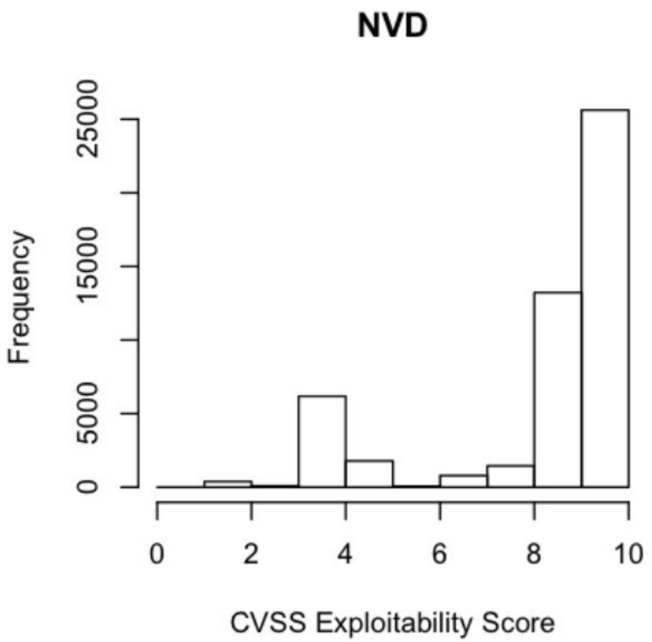


Image from He et al. 2016

The domain of data

- Finding large amount of data for every task we want our machine learning model to perform is hard.
    - e.g. doing machine translation on rare language pair.
    - e.g. data about cyber vulnerabilities exploits

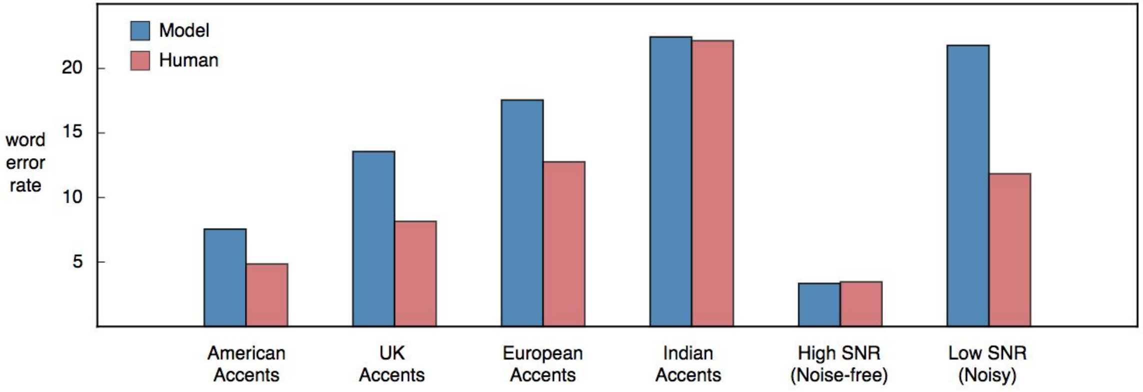


MACHINE INTELLIGENCE COMMUNITY

Image from Allodi and Massacci 2013

The domain of data

- Sometime our input data varies.
  - e.g. speech recognition is challenging because every person's speech is a little bit different. Furthermore, there are different accent of the same language, thus one model for one language would not work well.

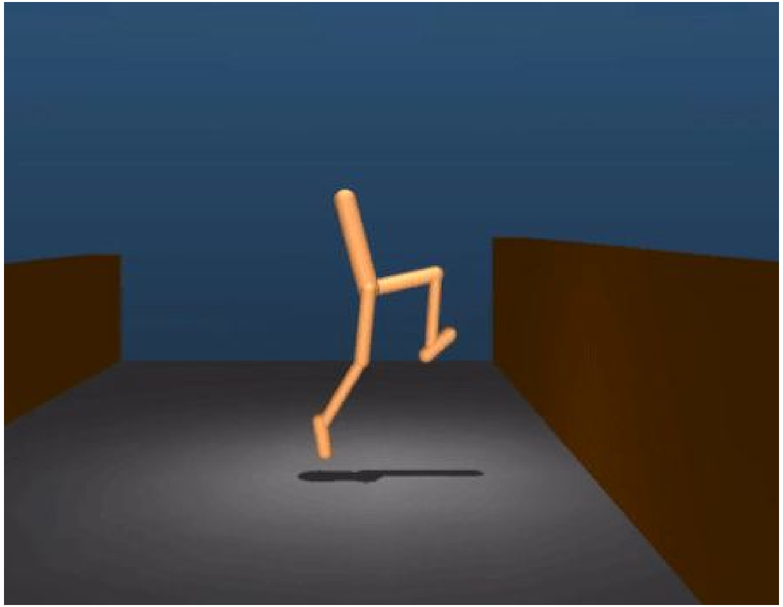


MACHINE INTELLIGENCE COMMUNITY

Chart from [Awni Hannun]

Learning from simulation

- Use a simulation of the real world to train a model.
  - Faster training time
  - Less costly
  - Able to repeat multiple time
- But the data from the simulation is still not the same data from the real world.
  - We need the ability to adapt the model from using the data in the simulated world to the data



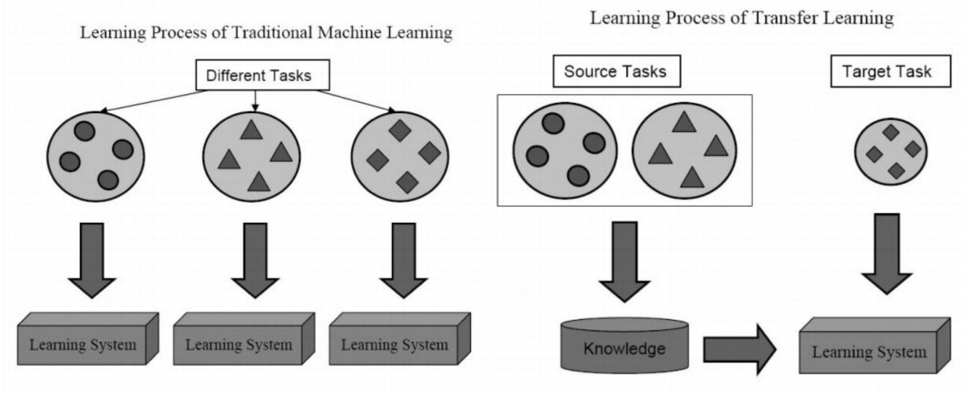
DeepMind's simulated environment

MACHINE INTELLIGENCE COMMUNITY

Image from DeepMind

So, transfer learning

- Utilise the feature extraction property of Convolutional Neural Networks
  - Reduce training time and computational cost.
  - Reduce the amount of data used.
- Improve performance with task where the data domain varies.
  - In general, instead of mapping data to output, we distill some “knowledge” about the data and use that to make a better machine learning model.



MACHINE INTELLIGENCE COMMUNITY

Image from Pan and Yang 2010

More formally

$$\mathcal{D} = \{\mathcal{X}, P(X)\}$$

Domain

Feature space

Marginal probability distribution

When $D_s \neq D_t$ or $T_s \neq T_t$, **transfer learning** aims to improve the target predictive function (the conditional probability distribution) in D_t using knowledge from D_s and T_s

Task

Output/Label space

$$\mathcal{T} = \{\mathcal{Y}, P(Y|X)\}$$

Conditional probability distribution
(this is what's usually learnt)

MACHINE INTELLIGENCE COMMUNITY

Different scenarios of transfer learning

The domains are different

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

The feature spaces of the data are different from each other.
E.g. different languages.

$$P(X_S) \neq P(X_T)$$

The marginal probability distributions of the data are different.
E.g. different topic of document.

The tasks are different

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

The label spaces are different.
E.g. classify the same document into different label depend on different task.

$$P(Y_S|X_S) \neq P(Y_T|X_T)$$

The learned functions are different.
E.g. the source and target distribution of the data is unbalanced compared to their class.

MACHINE INTELLIGENCE COMMUNITY

Different setting of transfer learning

- Both domain and task are different -> **Unsupervised Transfer Learning**.
- Domains are different -> **Transductive Transfer Learning**
- Tasks are different -> **Inductive Transfer Learning**

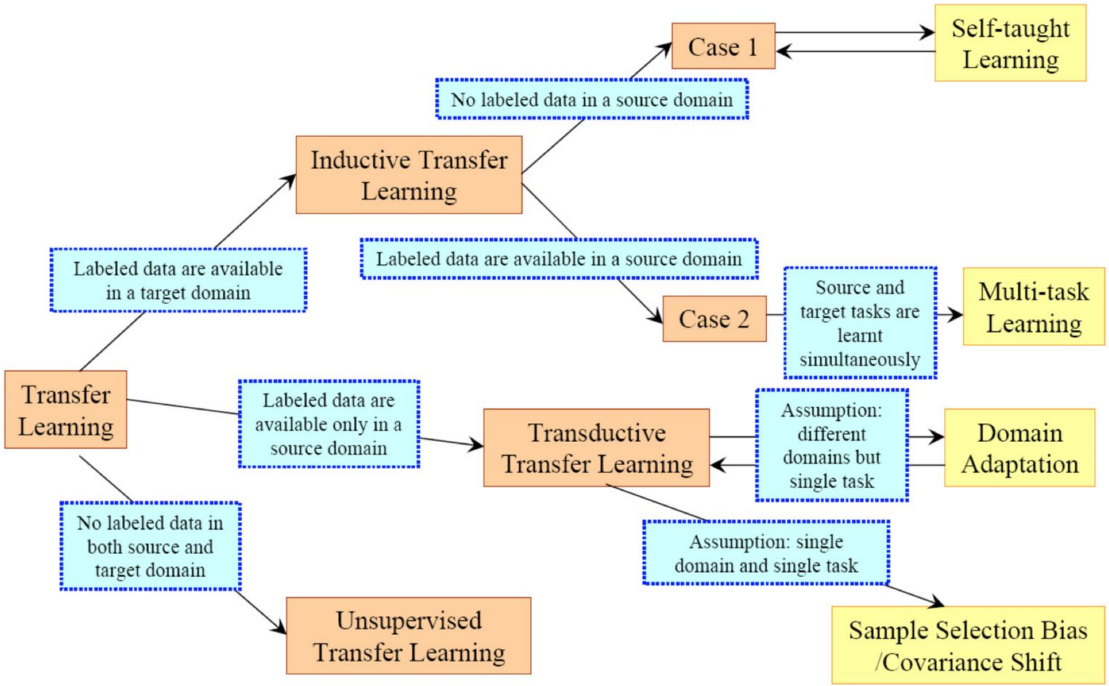
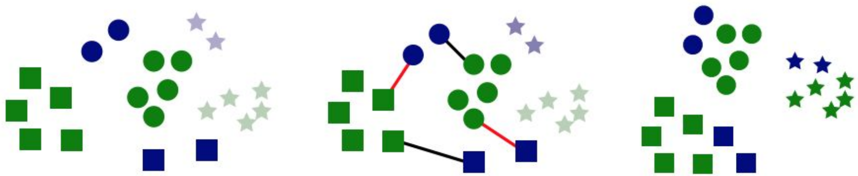


Image from [Pan and Yang 2010]

Domain Adaptation

- We are learning the same task but on different domain of data.
- To make our model adapt to the change in domain, we could make the two domain “closer” to each other.
    - Transform data from the domains into an invariant space (K. Saenko et al. 2010)



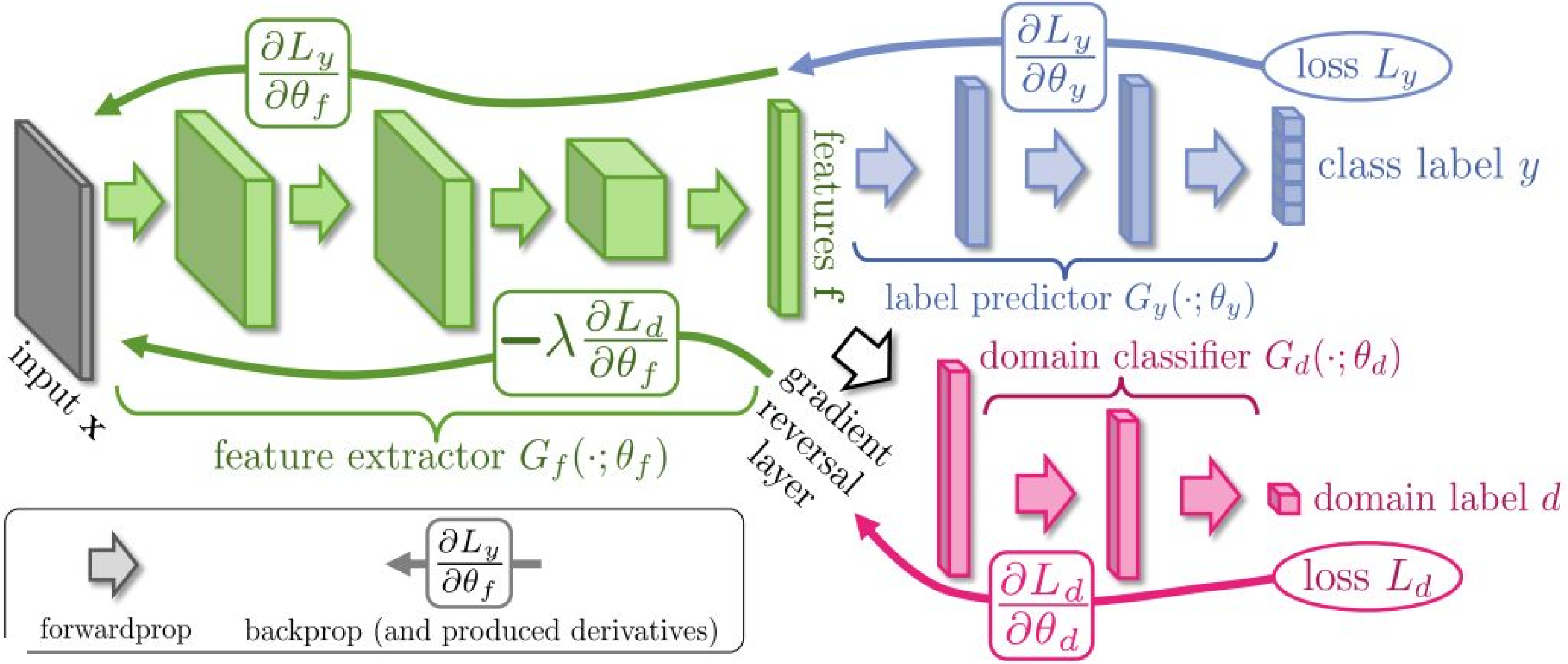
(a) Domain shift problem (b) Pairwise constraints (c) Invariant space

    - Make it an objective of the model to “confuse” the domains of the data (Ganin 2015)

MACHINE INTELLIGENCE COMMUNITY

Image from Saenko et al. 2010

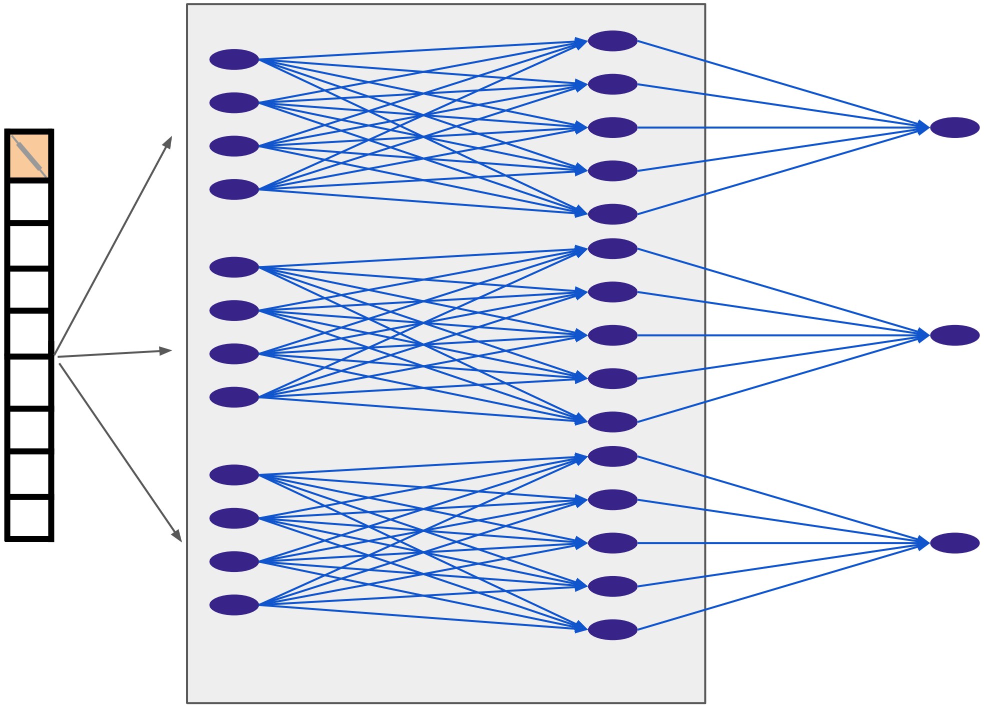
Domain confusion using backpropagation



MACHINE INTELLIGENCE COMMUNITY

Image from Ganin and Lempitsky 2015

Multi-task learning



- Instead of training a model for each task, let's train one model for all the task and share the parameters for the models.
- This take advantage of the hierarchical representation of the data through the network.

MACHINE INTELLIGENCE COMMUNITY

Few-shot, One-shot, Zero-shot learning

- Data is expensive. Can we use transfer learning to learn with less data?

- **One-shot** learning means that the model have one example of the class before it has to make the prediction and **zero-shot** learning is making prediction on brand new data.

MACHINE INTELLIGENCE COMMUNITY

Example of few shot learning in human

- If you are not familiar with this spiky fruit on the left here, it’s a durian.
- Human has the ability to learn/understand concepts, ideas, or objects from very small “training” example.

MACHINE INTELLIGENCE COMMUNITY

Image from Wikimedia Commons

Example of few shot learning in human

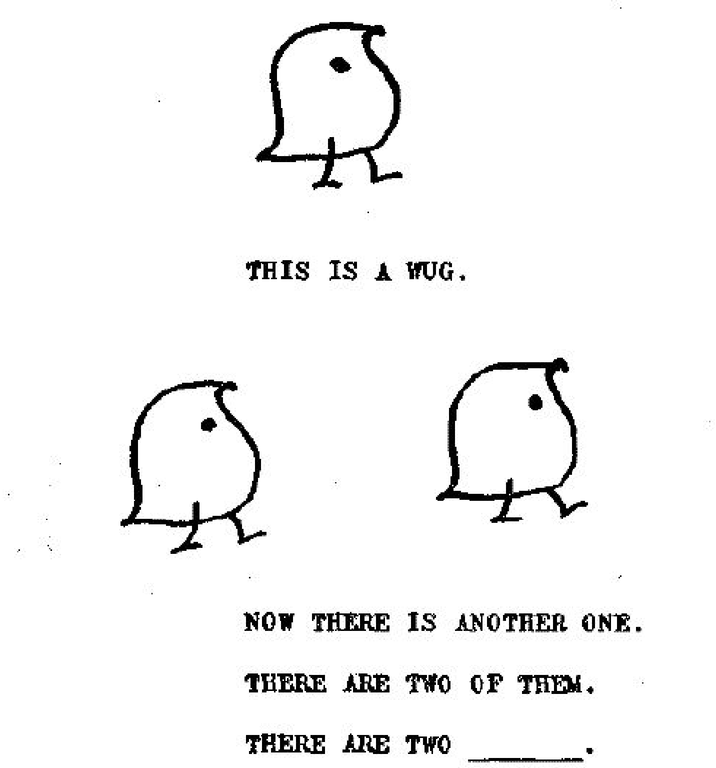
- So if you see the image of this spiky fruit again, you would easily be able to recognise that's it's a durian.
    - In essence, you only require one or few training example to "learn" new thing.

MACHINE INTELLIGENCE COMMUNITY

Image from Wikimedia Commons

Example of zero shot learning in human

- Furthermore, human can generalise from other knowledge.
  - The wug test developed by Berko Gleason (1958) show that very young child can apply morphological rule to unknown words.
  - Here, we don’t know the plural form of the word “wug”, but we are able to formulate it anyway.



MACHINE INTELLIGENCE COMMUNITY

Image from Berko 1958

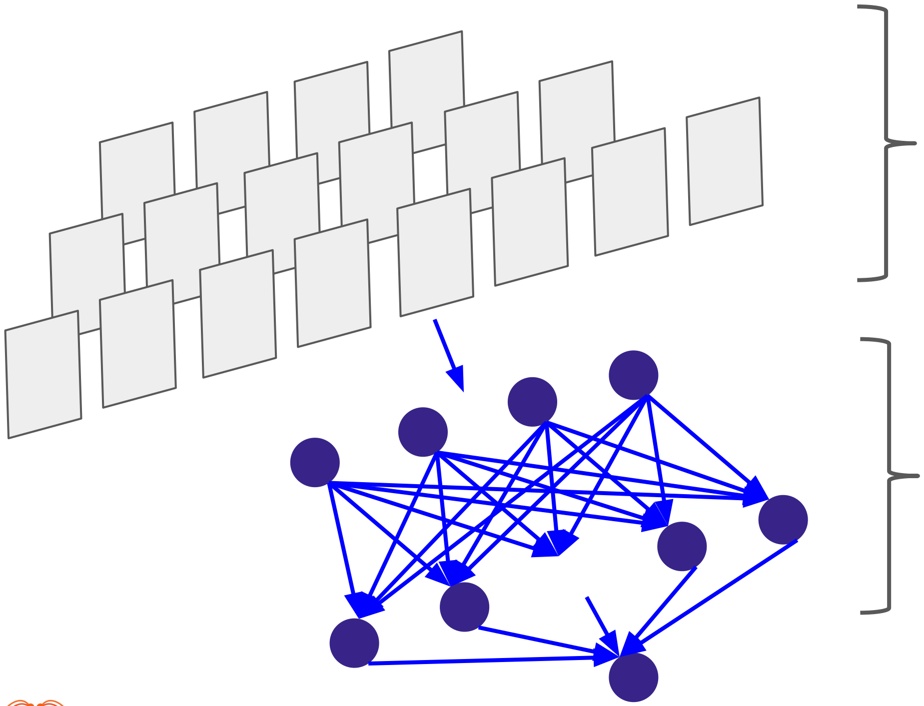
Word-embedding

- Turns words to vector representation.
- Models like *word2vec* exploit the fact that semantically similar words tend to be around similar words.
- Vector representation have semantics value.

Image from Olah 2014

MACHINE INTELLIGENCE COMMUNITY

Pre-training and fine-tuning



- **Fine-tuning** is when you have a different but related dataset and you want to train the whole model to better fit what you have.
- **Pre-train** models have all the weights already trained. Just make a new Fully Connected layer at the end and retrain the layer.

MACHINE INTELLIGENCE COMMUNITY

Pretrained models with PYTORCH

```python
import torchvision.models as models
resnet18 = models.resnet18(pretrained=True)
alexnet = models.alexnet(pretrained=True)
squeezenet = models.squeezenet1_0(pretrained=True)
vgg16 = models.vgg16(pretrained=True)
densenet = models.densenet161(pretrained=True)
inception = models.inception_v3(pretrained=True)
```

```python
for param in model_conv.parameters():
    param.requires_grad = False
```

```python
num_ftrs = model_conv.fc.in_features
model_conv.fc = nn.Linear(num_ftrs, 3)
```

- Import and instantiate pretrained model
- Remove the output layer of model
- Append a randomly initialized classifier and fine-tune

Try it out!

MACHINE INTELLIGENCE COMMUNITY

References & Further Reading

1. http://machinelearning.wustl.edu/mlpapers/paper_files/ICML2011Glorot_342.pdf
2. http://www.deeplearningbook.org/contents/representation.html
3. http://anthropology.uwo.ca/faculty/creider/027/wugs.pdf
4. https://www.cs.princeton.edu/~rajeshr/papers/cacm2011-researchHighlights-convDBN.pdf
5. http://ruder.io/transfer-learning
6. http://ieeexplore.ieee.org/document/5288526/

Shoutout to our Sponsors

Boston University Computer Science

Boston University SPARK!

Boston University Software Application and Innovation Lab

MACHINE INTELLIGENCE COMMUNITY

Upcoming Events

MIC Paper signup: https://goo.gl/iAm6TL
BUMIC Projects signup: https://goo.gl/GmP9oK

BUMIC paper discussion:

Paper: **Decoupled Neural Interfaces using Synthetic Gradients**
Location: Fishbowl Conference Room
Date: 11.06.17 Time: 7 PM

Next workshop:

Topic: Sequential Data
Location: BU Hariri Seminar Room
Date: 11.07.17 Time: 7 PM

MACHINE INTELLIGENCE COMMUNITY