Deep Learning - COSC2779

Representation Learning

Dr. Ruwan Tennakoon

RMIT UNIVERSITY

September 27, 2021

Outline

Problems in Applying Deep Learning

When applying deep learning in practice:

- Usually, **deep network** architectures are required for **complex tasks**.
- They have many parameters and **need large labeled datasets** to obtain good generalization.
- Generating large "labeled" dataset for your task is difficult.
  - **Time consuming**: You may want to complete the project withing tight deadlines.
  - **Expensive**: Annotating a large dataset is expensive (infrastructure and manpower).
  - **Not possible**: e.g. in some medical imaging tasks, it is not ethical to put a healthy person through imaging or prevalence can be low.

Deep learning can only be applied when large datasets are available.

Problems in Applying Deep Learning

When applying deep learning in practice:

- Usually, **deep network** architectures are required for **complex tasks**.
- They have many parameters and **need large labeled datasets** to obtain good generalization.
- Generating large "labeled" dataset for your task is difficult.
  - **Time consuming**: You may want to complete the project withing tight deadlines.
  - **Expensive**: Annotating a large dataset is expensive (infrastructure and manpower).
  - **Not possible**: e.g. in some medical imaging tasks, it is not ethical to put a healthy person through imaging or prevalence can be low.

~~Deep learning can only be applied when large labeled datasets are available.~~
Not correct! There are alternatives.

Objective of the lecture

Explore practical approaches used in deep learning that can handle limited data availability:

- Relatively small labeled dataset only: **Transfer learning**.
- Large unlabelled data: **Self supervised learning**.

Outline

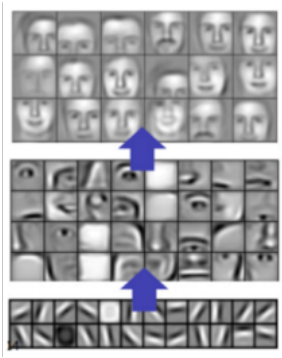


Image from: Paper

Learned features from a face recognition task.

Observation:

- Layers towards the bottom (close to input) learn *general features*.
- Layers towards the top (close to output) learn task *specific features*.

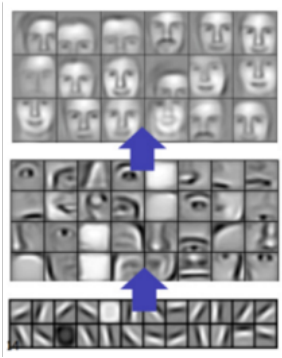


Image from: Paper

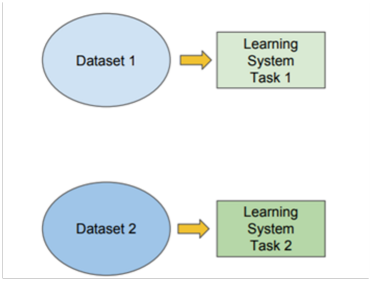
Learned features from a face recognition task.

Observation:

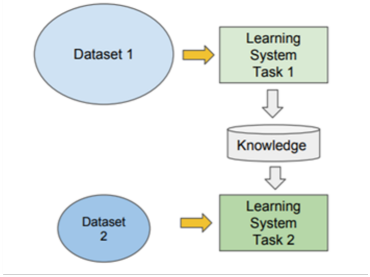
- Layers towards the bottom (close to input) learn *general features*.
- Layers towards the top (close to output) learn task *specific features*.

Can we take the learned features on one task and apply them to a new task?

Transfer Learning



Traditional



Transfer Learning

Outline

Assume you are designing a "Pet feeding machine" that dispenses the appropriate food type based on weather a cat or a dog is nearby.

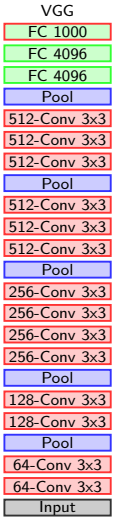
You have decided to use vision to identify if a cat or a dog is nearby.

Cats vs Dogs classification can be complex as there are many types of cats/dogs.

Can collect a small dataset with labels. Training dataset size: 2000 images from both classes.

- Find a very large dataset that has similar data, train a big CNN there (or take pre-trained models).
- Transfer what is learned to your dataset.

Why would this word? Features in the bottom layers are general. Knowledge gained will transfer to other tasks.

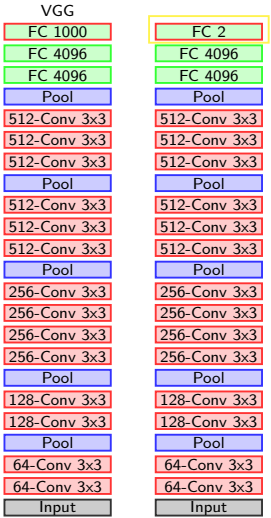


Step 1: Select a related task (proxy task) that has large dataset. e.g. ImageNet.

Step 2: Train a selected base model on the proxy task or download the pre-trained model. e.g. VGG/ResNet.

We can assume that the top most layer is very specific to the ImageNet task (not very related to our cats vs dog). The bottom layers are general.

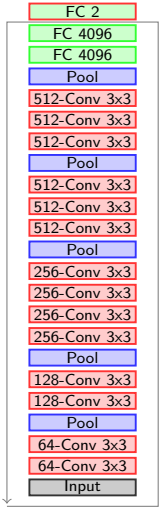
∗ This is not the best example because cats and dogs are categories in ImageNet. The process will work even when the categories are not directly in ImageNet (E.g. Gaze direction classification in face images.)



Step 3: Identify the feature layer you want to use for your task. Usually the **bottleneck layer** (the one before the final classification layer).

Step 4: Remove the layers above the bottleneck layer and reconfigure to your task.

- New top part is called the **head**.
- Remaining parts from the pre-trained model is called the **base model**.



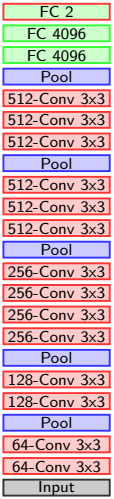
Step 5: Freeze all the layer below (and including) bottleneck layer (weights in base model).

Step 6: Train the network on the new task. E.g Cats vs Dogs.

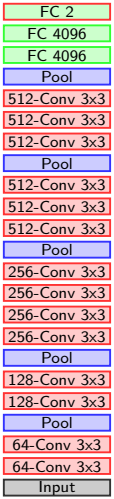
Step 7: (optional) Fine-tuning; Unfreeze "some" top layers in the frozen base model and train the network for some more epoch on the new task. This will train both head and base model weights.

Now you have a trained model for the task. We will do an example in the lab.

Important Considerations



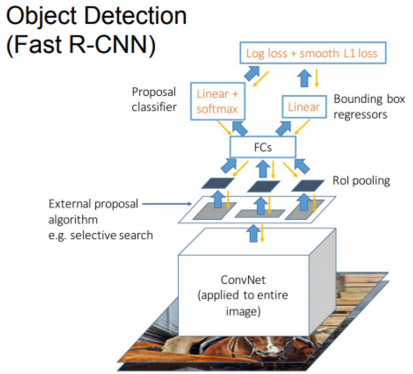
- The **input normalization and pre-processing** has to match with the techniques employed in training the original model.
- Should use a **smaller learning rate when fine-tuning** to avoid forgetting the pre-learned features. Over-fitting to new task.
- Should not train the head and the base model weights together without first tuning the head with frozen base model.



	Similar datasets	**Different datasets**
Relatively Small Dataset	Train the classifier on top	You are in trouble. Get more data
Relatively Large dataset	Fine tune Few more layers	Fine tune a larger number of layers

Transfer learning for images: summary

1. Select a related proxy task that has large dataset (or pre-trained model).
2. Train a selected base model on the proxy task or download the pre-trained model.
3. Identify the feature layer you want to use for your task.
4. Remove the layers above the bottleneck layer and reconfigure to your task (attach a head).
5. Freeze all the layer in base model.
6. Train the network on the new task.
7. (optional) Unfreeze some top layers in the base model and fine-tune.



Object Detection

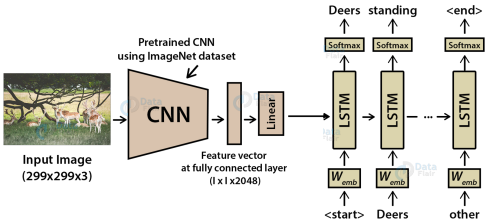


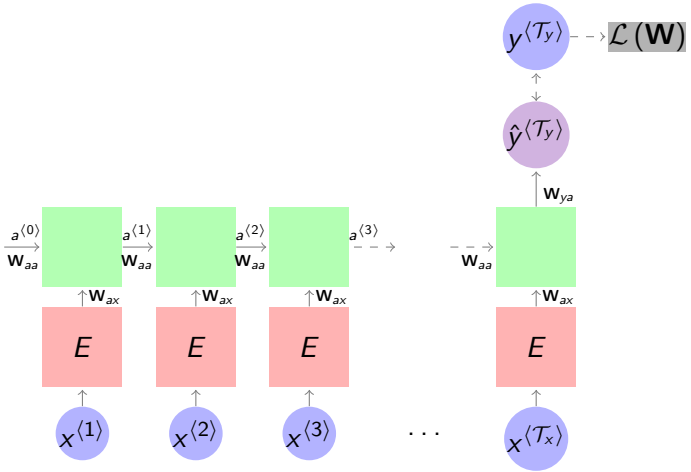
Image Captioning

Nowadays Transfer learning is the norm.

Outline

Assume you are hired by RMIT to design a system that automatically mark an email coming into the inbox of RMIT academics as: “Urgent”, “Non-Urgent”.

The training dataset is 2000 emails with correct annotation.

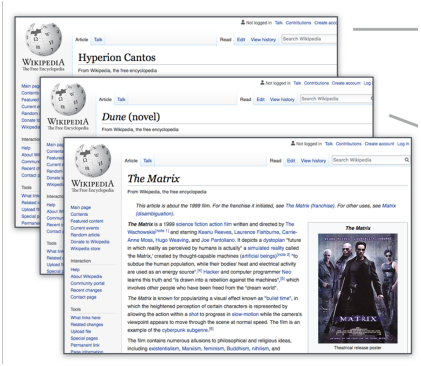


Need an RNN to capture the information in the language structure.

Not enough examples to learn the complexities and semantics of language.

- Find a very large dataset that has **similar** language structure, train a word embedding matrix there (or take pre-trained models).
- Transfer what is learned to your dataset.

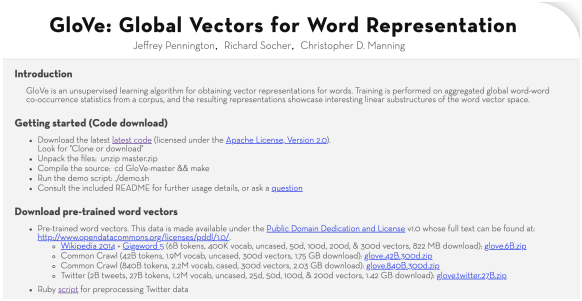
Why would this word? Word embeddings learn the semantics in the language. The knowledge gained will transfer to other tasks.



Option 1:

Step 1.1: Find a large text corpus related to the task. e.g. Spam/no-spam email dataset, Wikepedia, Twitter.

Step 1.2: Learn embedding matrix, E, using techniques discussed last week. e.g. language modelling (predict next word).



GloVe: Global Vectors for Word Representation

Jeffrey Pennington, Richard Socher, Christopher D. Manning

Introduction

GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space.

Getting started (Code download)

- Download the latest latest code (licensed under the Apache License, Version 2.0). Look for "Clone or download".
- Unpack the files: unzip master.zip
- Compile the source: cd GloVe-master && make
- Run the demo script: ./demo.sh
- Consult the included README for further usage details, or ask a question

Download pre-trained word vectors

- Pre-trained word vectors. This data is made available under the Public Domain Dedication and License v1.0 whose full text can be found at: http://www.opendatacommons.org/licenses/pddl/1.0/.
  - Wikipedia 2014 + Gigaword 5 (6B tokens, 400K vocab, uncased, 50d, 100d, 200d, & 300d vectors, 822 MB download): glove.6B.zip
  - Common Crawl (42B tokens, 1.9M vocab, uncased, 300d vectors, 1.75 GB download): glove.42B.300d.zip
  - Common Crawl (840B tokens, 2.2M vocab, cased, 300d vectors, 2.03 GB download): glove.840B.300d.zip
  - Twitter (2B tweets, 27B tokens, 1.2M vocab, uncased, 25d, 50d, 100d, & 200d vectors, 1.42 GB download): glove.twitter.27B.zip
- Ruby script for preprocessing Twitter data

Option 2:

Step 1: Download pre-learned word embeddings.

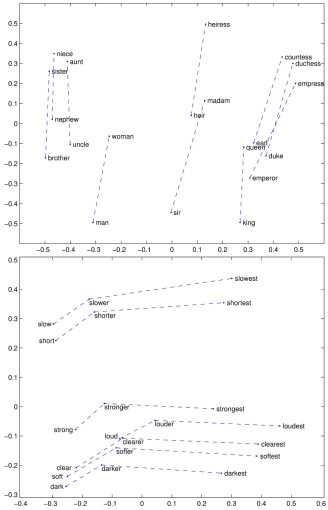
- Word2Vec
- GloVe

Option 2 is the most popular.

Pre-trained embedding matrices (E_{glove}) available as `.txt` files.

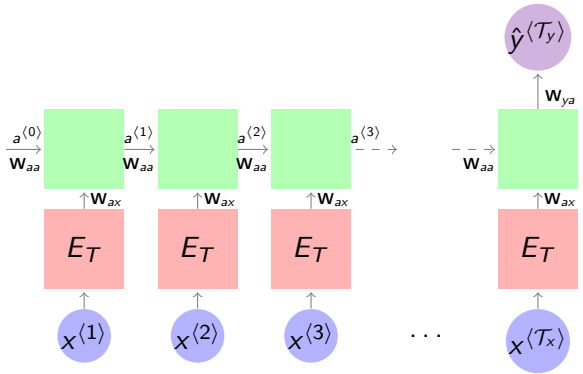
This data is made available under the Public Domain Dedication and License v1.0

- Wikipedia 2014 + Gigaword 5 (400K vocab, uncased, 50d, 100d, 200d, & 300d vectors)
- Common Crawl (1.9M vocab, uncased, 300d vectors)
- Common Crawl (2.2M vocab, cased, 300d vectors)
- Twitter (1.2M vocab, uncased, 25d, 50d, 100d, & 200d vectors)



Step 2: Map the vocabulary of word embedding to vocabulary of the task.

1. Tokenize the training set of the task to get the task-vocabulary or select a suitable vocabulary.
2. Map each word of the task-vocabulary to embedding-vocabulary.
3. Compose the embedding matrix (E_T) that has number of rows equal to the number of words in the task-vocabulary.



Step 3: Plug in the E_T to the embedding layer.

Step 4: Freeze the Embedding layer and train the RNN part.

Step 5: (optional) Fine-tune the embedding. Unfreeze embedding layer and train for few more epoch with a smaller learning rate.

- The Illustrated BERT, ELMo, and co. - How NLP Cracked Transfer Learning
- Efficient multi-lingual language model fine-tuning

Outline

To some extent, any visual task can be solved now by:

- Construct a large-scale dataset labelled for that task.
- Specify a training loss and neural network architecture.
- Train the network and deploy.

Strong Supervision

To some extent, any visual task can be solved now by:

- Construct a large-scale dataset labelled for that task.
- Specify a training loss and neural network architecture.
- Train the network and deploy.

Issues:

- Expense of producing a new dataset for each new task.
- Some areas are supervision-starved, e.g. medical data, where it is hard to obtain annotation.
- Untapped/availability of vast numbers of unlabelled images/videos.

Outline

Medical Image Classification

Assume you are hired by a hospital to design a system to detect prostate cancer in CT images.

The hospital has provided you with a small labelled dataset of 600 CT scans (both benign and malignant).

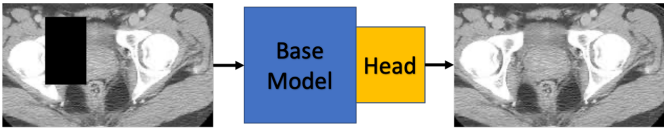
The hospital has also given you access to their imaging archives which had **many more CT images**. However these images are **not labelled** and it will take considerable resources to annotate them (benign and malignant).

Not enough images to train from scratch.

Transfer learning is not an option here are we do not have large dataset from a similar task (supervised) and no pre-trained models.

- **Step 1**: Design a proxy task that does not require manual labelling.
- **Step 2**: Train a base model on the proxy task.
- **Step 3**: Do transfer learning with small labelled dataset.

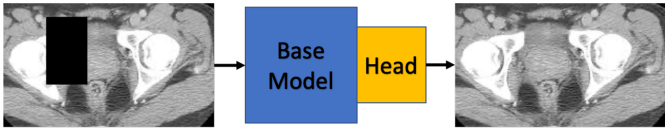
The proxy task has to be designed in a way that the model will capture information relevant to the task. This involved lots of engineering and we will not go into detail in this course.



Proxy Task: Randomly blackout a small region in a CT scan and try to recreate the original scan (with no blackout).

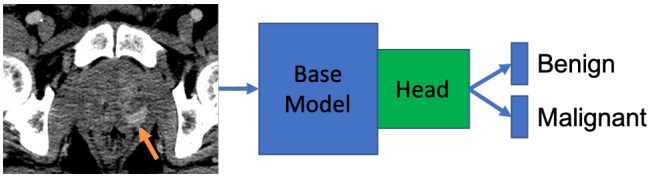
The unlabeled dataset can now be used to create a labeled dataset for the proxy task automatically (at no expense).

This proxy task is called “inpainting”.



Proxy Task: Randomly blackout a small region in a CT scan and try to recreate the original scan (with no blackout).

Intuition: To do the proxy task the network must learn about the structure of the CT scans.



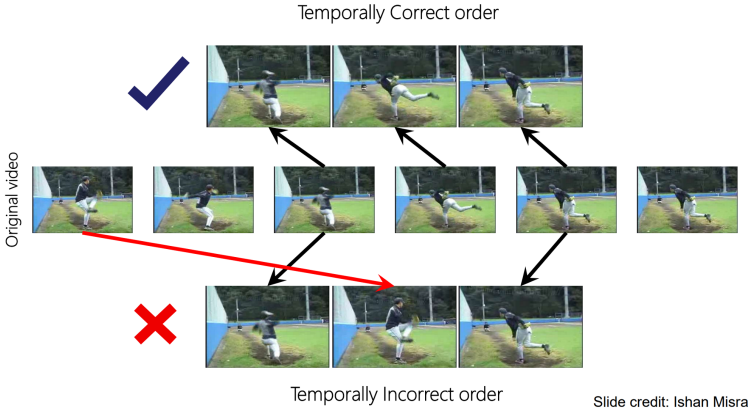
Intended Task: Transfer and Train the model with small supervised dataset. Fine tune if necessary.

- A form of unsupervised learning where the data provides the supervision.
- In general, withhold some information about the data, and task the network with predicting it.
- The task defines a proxy loss, and the network is forced to learn what we really care about, e.g. a semantic representation, in order to solve it.

Many self-supervised tasks for images, videos.

Which image has the correct rotation?

Unsupervised Representation Learning by Predicting Image Rotations



Slide credit: Ishan Misra

Shuffle and Learn: Unsupervised Learning using Temporal Order Verification

Outline

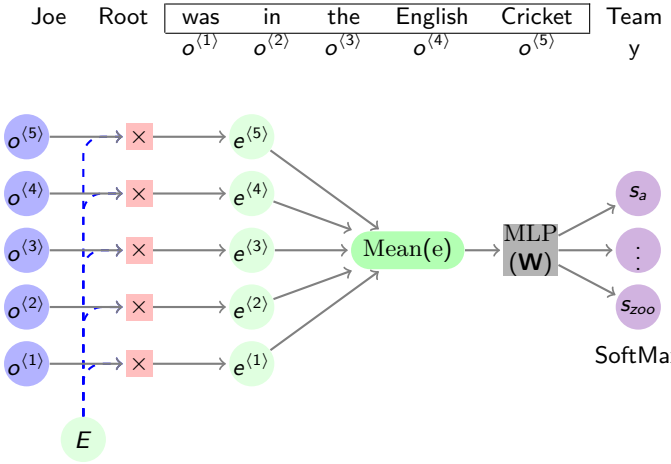
Learn by language modelling. E.g. Predict Next word:

Joe Root was in the English cricket ???.

Can create a supervised learning task where **x** is a fixed length sequence (window selected by user) and y is the next word in the sequence.

Joe	Root	was	in	the	English	Cricket	Team
		$x^{\langle 1 \rangle}$	$x^{\langle 2 \rangle}$	$x^{\langle 3 \rangle}$	$x^{\langle 4 \rangle}$	$x^{\langle 5 \rangle}$	y

A Neural Probabilistic Language Model



Train for parameters E and the weights of the MLP ($\mathbf{W}$) via back-propagation.

Joe	Root	was	in	the	English	Cricket	Team
		$x^{\langle 1 \rangle}$	$x^{\langle 2 \rangle}$	$x^{\langle 3 \rangle}$	$x^{\langle 4 \rangle}$	$x^{\langle 5 \rangle}$	y
				context			target

Context can be:

- Last 5 words.
- 5 words before and 5 words after.
- Last one word.
- Near by word.

Word2Vec and GloVe are more recent practical word embedding learning techniques.

Yann Lecun's Cake Theory at NIPS 2016

- "Pure" Reinforcement Learning (cherry)
  - The machine predicts a scalar reward given once in a while.
  - **A few bits for some samples**
- Supervised Learning (icing)
  - The machine predicts a category or a few numbers for each input
  - Predicting human-supplied data
  - **10→10,000 bits per sample**
- Unsupervised/Predictive Learning (cake)
  - The machine predicts any part of its input for any observed part.
  - Predicts future frames in videos
  - **Millions of bits per sample**

- (Yes, I know, this picture is slightly offensive to RL folks. But I'll make it up)

Yann LeCun is a co-recipient of the **2018 Turing Award** for his work in deep learning.

Summary

How to build deep model when we do not have large labeled datasets.

- Transfer learning: Need pre-trained model on "similar" dataset + small amount of labeled data.
- Self-supervision: Need large unlabeled dataset.

Next week: Unsupervised learning.

Lab: Transfer Learning
Tutorial: Transfer Learning (Lab for week 10)