The Unreasonable Effectiveness of Transfer Learning on NLP

David Low

Pand.ai
Bio

- **Research**
  - Urban Mobility | Social Media

- **Public Service**
  - GovTech(IDA) Data Science Division

- **Teach**
  - Adjunct Lecturer at National University of Singapore (NUS)

- **Startup**
  - Conversational AI

- **Technical Reviewer**
  - Packt Publications, UK
  - Manning Publications Co., US
Overview

- Transfer Learning
- ImageNet and Feature Hierarchy
- Approaches and Considerations
- Previous attempts in NLP
- Recent advancements
- Language Modeling
- ULMFiT: Universal Language Model for Fine-Tuning
- Code Walkthrough
- Resources
What is Transfer Learning?

- Transfer learning is a concept where we try to leverage knowledge learned previously to solve new problems.
- For example, learning to play one music instrument can facilitate faster learning of another music instrument.
- Transfer learning has gained attention since its discussion in the Neural Information Processing Systems 1995 workshop on "Learning to Learn".
Overview of Different Settings of Transfer Learning

- **Self-taught Learning**
  - Case 1: No labeled data in a source domain
  - Inductive Transfer Learning
    - Labeled data are available in a target domain
  - Case 2: Source and target tasks are learnt simultaneously
    - Multi-task Learning
      - Labeled data are available in a source domain
    - Domain Adaptation
      - Labeled data are available only in a source domain
    - Sample Selection Bias / Covariance Shift
      - Assumption: single domain and single task
      - Assumption: different domains but single task

Source: Pan et al
Published in 2009
- 1.3 million images with 1,000 object classes
- ImageNet Large Scale Visual Recognition Challenge (2010 to 2017)
- AlexNet in 2012, 41% better than 2nd place.
- The beginning of Deep Learning era
Deep Neural Network
Features learned in each layer
Transfer Learning Approaches

Source: Yamashita et al 2018
Fine-tuning Considerations

Quadrant 1:
- Large dataset, but different from the pre-trained model's dataset.

Quadrant 2:
- Large dataset and similar to the pre-trained model's dataset.

Quadrant 3:
- Small dataset and different from the pre-trained model's dataset.

Quadrant 4:
- Small dataset and similar to the pre-trained model's dataset.

- Train the entire model.
- Train some layers and leave others frozen.
- Freeze the convolutional base.
How well do pre-trained ImageNet models generalize?

Object Detection

Semantic Segmentation

Human Pose Estimation

Human Action Classification
From Computer Vision to NLP

- *Is there a ImageNet-like dataset for natural language?*

  - **Data size**
    - On the order of millions of training examples.

  - **Representative of the problem space**
    - Allows us to learn most of the knowledge / relations required for understanding natural language

  - **Annotations**
    - Good quality labels
Earlier attempts of Transfer Learning on NLP

- **Word embedding models**
  - Word2vec (Mikolov et al 2013)
    - Based on distributional hypothesis: Words with similar meanings tend to occur in similar context.
  - GloVe: Global Vectors for Word Representation (Pennington et al 2014)
    - Word co-occurrence count-based approach
These embeddings have proven to be efficient in capturing context similarity and analogies.

They are fast and efficient due to its smaller dimensionality.
Shortcomings of shallow pre-training II

The GloVe word embedding of the word "stick" - a vector of 200 floats (rounded to two decimals).

Source: jalammar.github.io

Source: Dictionary.com
Shortcomings of shallow pre-training II

- Word2vec, GloVe and related methods are *shallow* approaches that trade expressivity for efficiency.
- Using word embeddings is like initializing a computer vision model with pretrained representations that only encode edges, missing the higher-level information required for downstream tasks.
- A model initialized with word embeddings needs to learn from scratch not only to disambiguate words, but also to model complex language phenomena such as long-term dependencies, agreement, negation, and many more.
- Hence, NLP models initialized with these shallow representations still require a huge number of examples to achieve good performance.
Recent Breakthroughs in NLP

- **ELMo** *(Peters et al 2018)*
  - “Deep contextualized word representations”

- **ULMFiT** *(Howard et al 2018)*
  - “Universal Language Model Fine-tuning for Text Classification”

- **OpenAI Transformer** *(Radford et al 2018)*
  - “Improving Language Understanding by Generative Pre-Training”
  - 12 layers, 8 GPUs, 1 month

- **BERT** *(Devlin et al 2018)*
  - “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”
  - 24 layers, 64 TPUs, 4 days. (8 GPUs, 40 - 70 days)
What is a language model?

- Generally, a Language Model is a model which is able to predict the next word, based on the sequence of words already seen.

Language modeling is chosen as the pre-training objective as it is widely considered to incorporate multiple traits of natural language understanding and generation.

A good language model requires learning complex characteristics of language involving syntactical properties and also semantical coherence.

- Example: “The service was poor, but the food was _____”
  - Ability to associate attributes used to describe food.
  - Ability to identify that the conjunction “but” introduces a contrast.

Training of a language model does not require any manual labeling and is considered as unsupervised / weakly-supervised.
ELMo: Contextualized Word Embeddings

- ELMo (Embeddings from Language Models)
- Instead of using a fixed embedding for each word, ELMo looks at the entire sentence before assigning each word in it an embedding. It uses a bi-directional LSTM trained on a specific task to be able to create those embeddings.

Source: jalammar.github.io

“Deep contextualized word representations” by Peters et al 2018
Achieve SOTA Performance across 6 challenging tasks

- Textual Entailment
- Named Entity Recognition
- Question Answering
- Coreference Resolution
- Semantic Role Labeling
- Sentiment Classification
ULMFiT: Universal Language Model for Fine-Tuning

- Proposed by Jeremy Howard and Sebastian Ruder in 2018 as a way to go a step further in transfer learning for NLP.
- The idea is to use a pre-trained language model (on a very large corpus of text, eg: a Wikipedia dump) and use it as a backbone/encoder for any downstream tasks.
3 Stages in ULMFiT

- **General Domain language model pre-training**
  - Language model pre-trained on Wikitext-103 (Merity et al., 2017). It consists of 28,595 pre-processed English Wikipedia articles and 103 million words.
  - AWD-LSTM (“Regularizing and Optimizing LSTM Language Models, Merity et al 2017)

- **Target task language model fine-tuning**
  - Fine-tune the pre-trained language model on data from the target task (on which classification will be performed).
  - The target text has a different distribution than the one on which our language model has been pre-trained.
  - Adjust the model weights such that they adapt to the task-specific text features. This step improves the performance of the downstream application, especially on small datasets.

- **Target task classifier training**
Bag of Tricks I

- Slanted Triangular Learning Rates (STLR)
  - STLR is a modification of the triangular learning rates (Smith et al 2017) with a short increase and a long decay period.
  - Model will quickly converge to a suitable region of the parameter space for the target task. Followed by a long decay period which allows for the further refining of the parameters.
Bag of Tricks II

- Discriminative fine-tuning
  - Different layers in a model capture different types of information and hence require different learning rate. The initial layers capture the most general form of information.
  - General information of the language are common and would require the least changes in their weights. The amount of fine-tuning required increases gradually as we move towards the last layer.
  - It first chooses the learning rate of the last layer by fine-tuning only the last layer and uses the following formula for the lower layers
    - \( \eta^{l-1} = \eta^l \times 0.3846 \), where \( \eta^l \) is the learning rate of the \( l \)-th layer.
Bag of Tricks III

- Gradual unfreezing
  - Gradually unfreeze the layers starting from the last layer to prevent catastrophic forgetting
  - When it comes to downstream task (classifier), an aggressive fine-tuning may erase the benefits of language model pre-training.

- How
  - The last LSTM layer is first unfrozen and the model is fine-tuned for one epoch.
  - Then the next lower frozen layer is unfrozen.
  - Repeats for all layers.
ULMFiT on real-world dataset

- Sentiment classification on IMDB dataset
- With only 100 examples + fine-tuning on pre-trained model, the performance is equivalent to the model trained from scratch with 20,000 examples!
Sentiment Classification on Amazon Review Dataset

- Inspired by the work done by Peter Martigny and his team from Feedly
- Results
  - Even with 50 samples only, they achieved 85% accuracy.
  - ULMFiT beats the reported score from FastText (~92%) with just 1000 samples.
  - Note that the reported score from FastText was using all 3.6M training samples.
- Based on Fastai v0.7 and PyTorch 0.4
Code Walkthrough

- Sentiment Classification on Amazon Review Dataset
- Ported to Fastai v1 library, compatible with PyTorch v1 and CUDA 10
- Code to be shared on Github: https://github.com/davidlowjw/strata_london_talk_ulmfit
“BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding” by Devlin et al 2018

https://openai.com/blog/better-language-models/ - OpenAI GPT-2

https://fast.ai - Making neural nets uncool again

http://ruder.io - Sebastian Ruder’s blog
Practice on a Kaggle competition

- An extension to “Toxic Comment Classification Challenge” last year.
- Build a model that detects toxicity and minimizes unintended bias associated with mentions of certain identities.
Paradigm Shift

- What we have witnessed in 2018
  - Paradigm shift from pre-trained Word Embeddings to Language Models
  - From just initializing the first layer of our models to pretraining the entire model with hierarchical representations.
  - If learning word vectors is like only learning edges, these approaches are like learning the full hierarchy of features, from edges to shapes to high-level semantic concepts.

- Bring us a step closer to Natural Language Understanding

- Looking forward to more exciting developments in the next few years!
LinkedIn: David Low https://www.linkedin.com/in/davidlowjw/
Twitter: @davidlowjw