Transfer learning in NLP

Transfer learning in CV

Transfer Learning is the reuse of a pre-trained model on a new problem.

Why is it cool?

- Faster convergence
- Less data for a new task

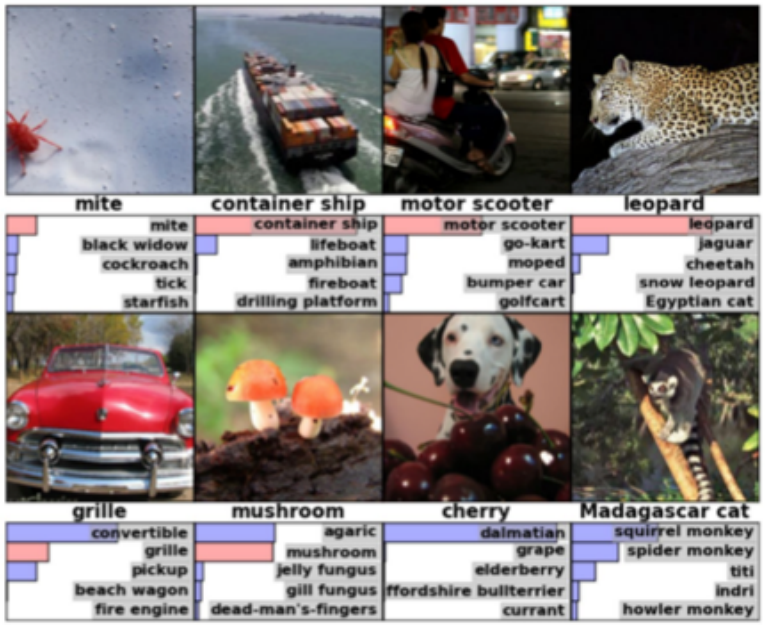
Popular architectures:

- AlexNet
- VGG19
- ResNet10
- ResNet50
- GoogleNet
- Etc.

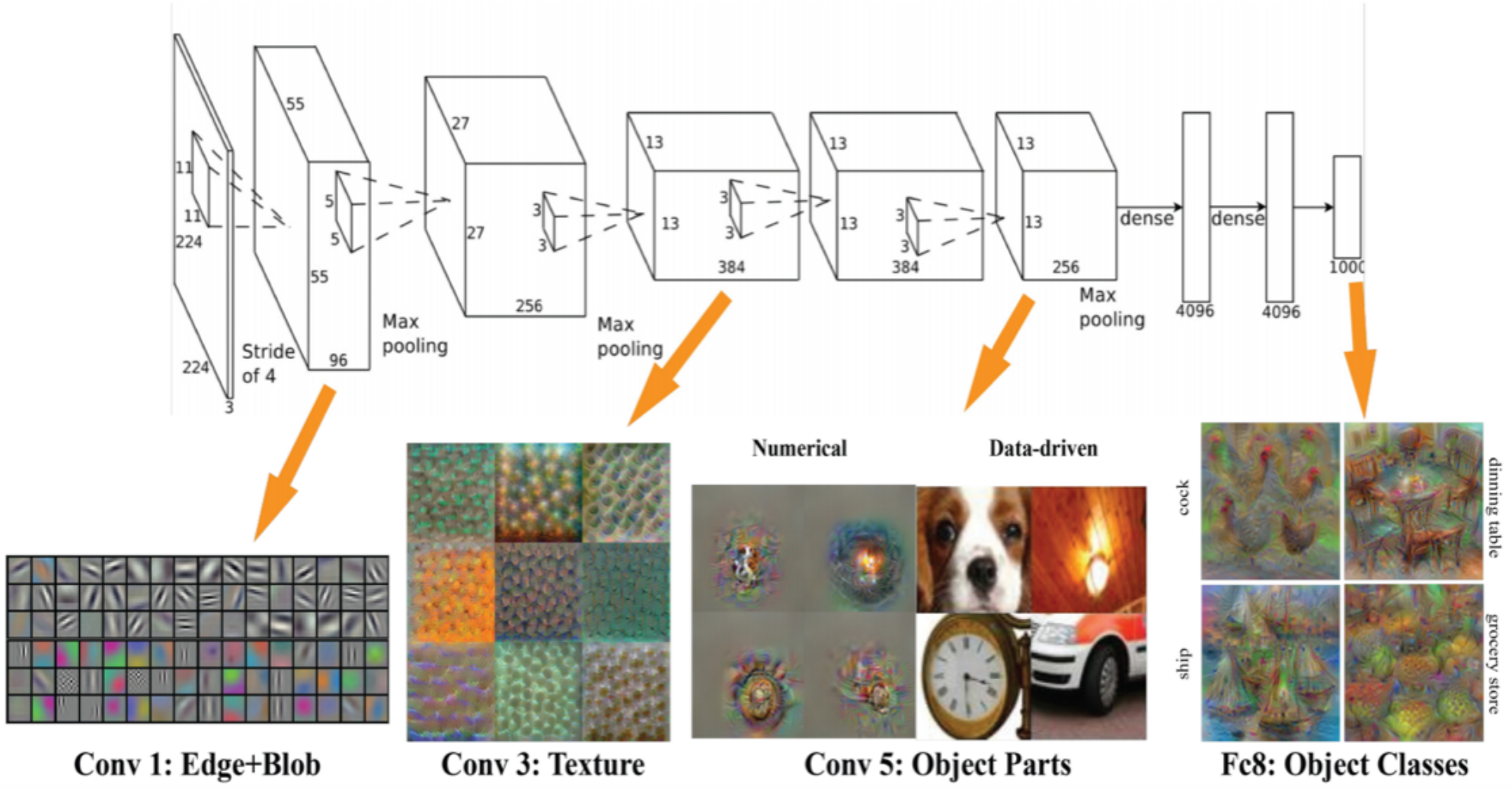
ImageNet Challenge

IMAGENET

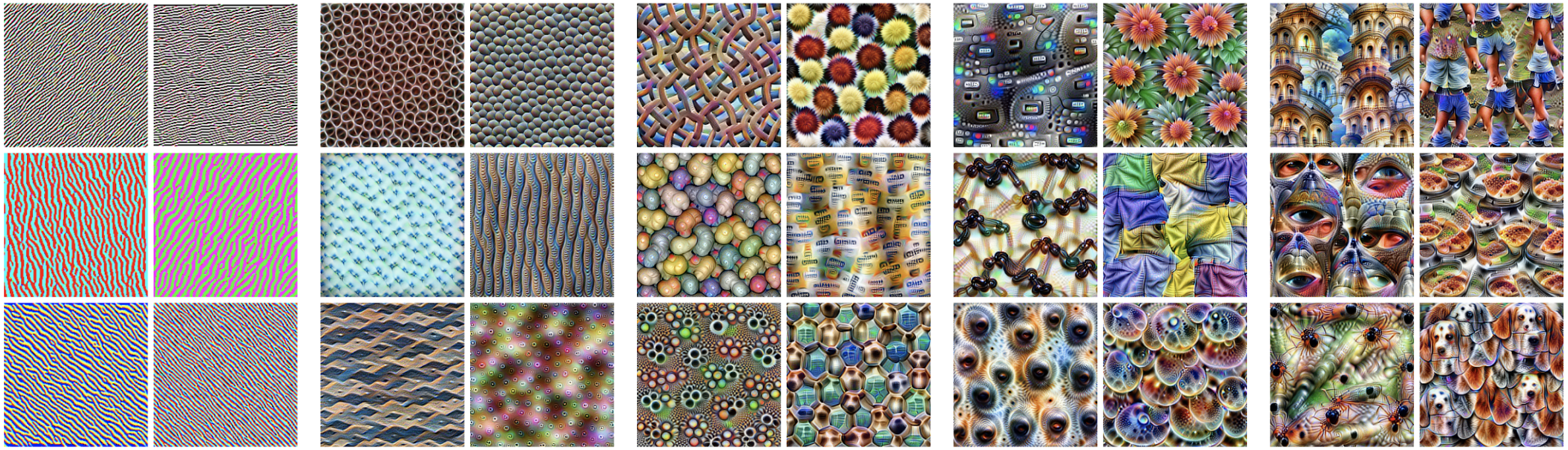
- 1,000 object classes (categories).
- Images:
  - 1.2 M train
  - 100k test.



Feature learning (1)



Feature learning (2)



Visualization of the information captured by features across different layers in GoogLeNet trained on ImageNet

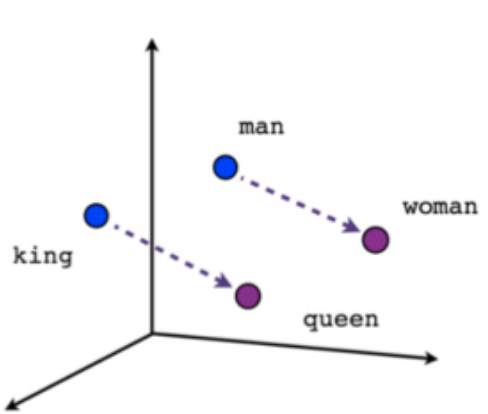
What about NLP?

- Still stuck on word embeddings
- Single-task models
- Different languages
- Most SOTA NLP results are obtained by training end-to-end architectures for each language task.
- Most datasets are in English, and are very small
- Not much knowledge sharing, or re-use
- Single task learning inhibits us from general language understanding, and exposes us to learning overly discriminative features that generalize poorly

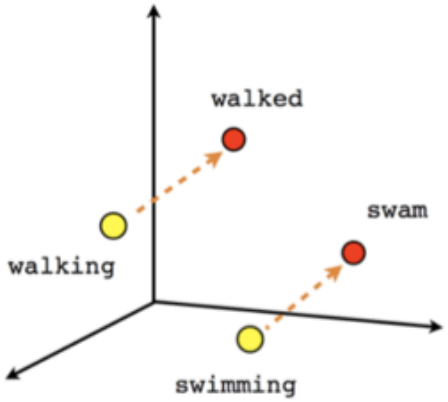
How to incorporate previous knowledge in NLP?

Popular methods:

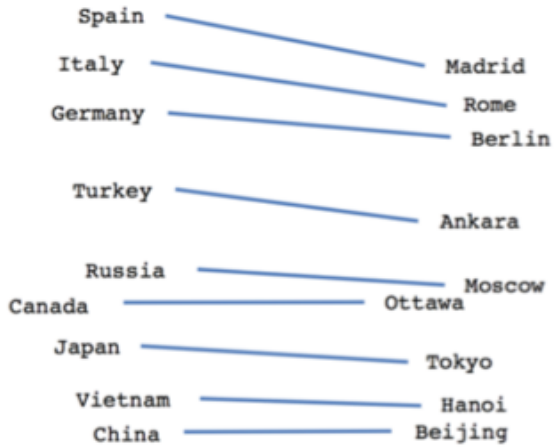
- Word2vec
- Fasttext
- Glove
- Etc.



Male-Female



Verb tense



Country-Capital

Why not as good?

- Only first layer initialization
- Context independent

- Models has to derive meaning from a sequence of words
- Models has to be able of modeling complex language phenomena such as compositionality, polysemy, anaphora, long-term dependencies, agreement, negation, and many more.

Solution: Language Modeling

- Goal: compute the probability of a sentence or sequence of words:

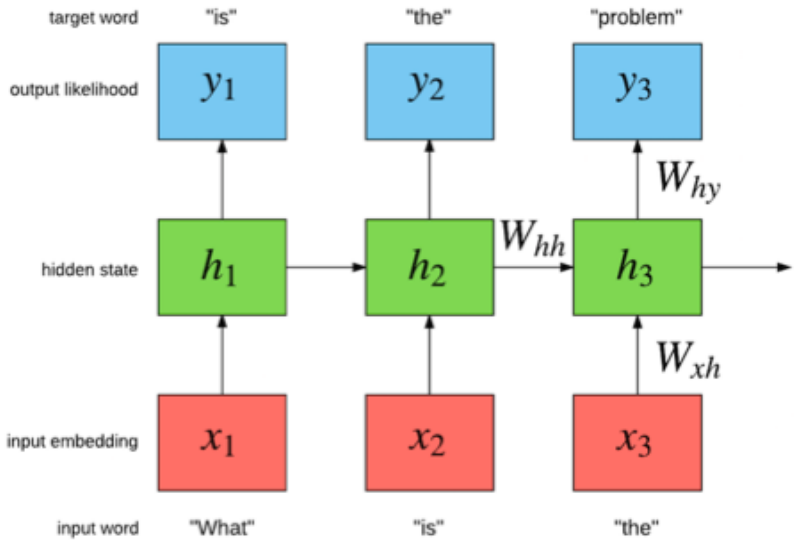
  $P(W) = P(w_1,w_2,w_3,w_4,w_5...w_n)$

- Related task: probability of an upcoming word:

  $P(w_5|w_1,w_2,w_3,w_4)$

- A model that computes either of these:

  $P(W)$ or $P(w_n|w_1,w_2...w_{n-1})$ is called a **language model**.



Why?

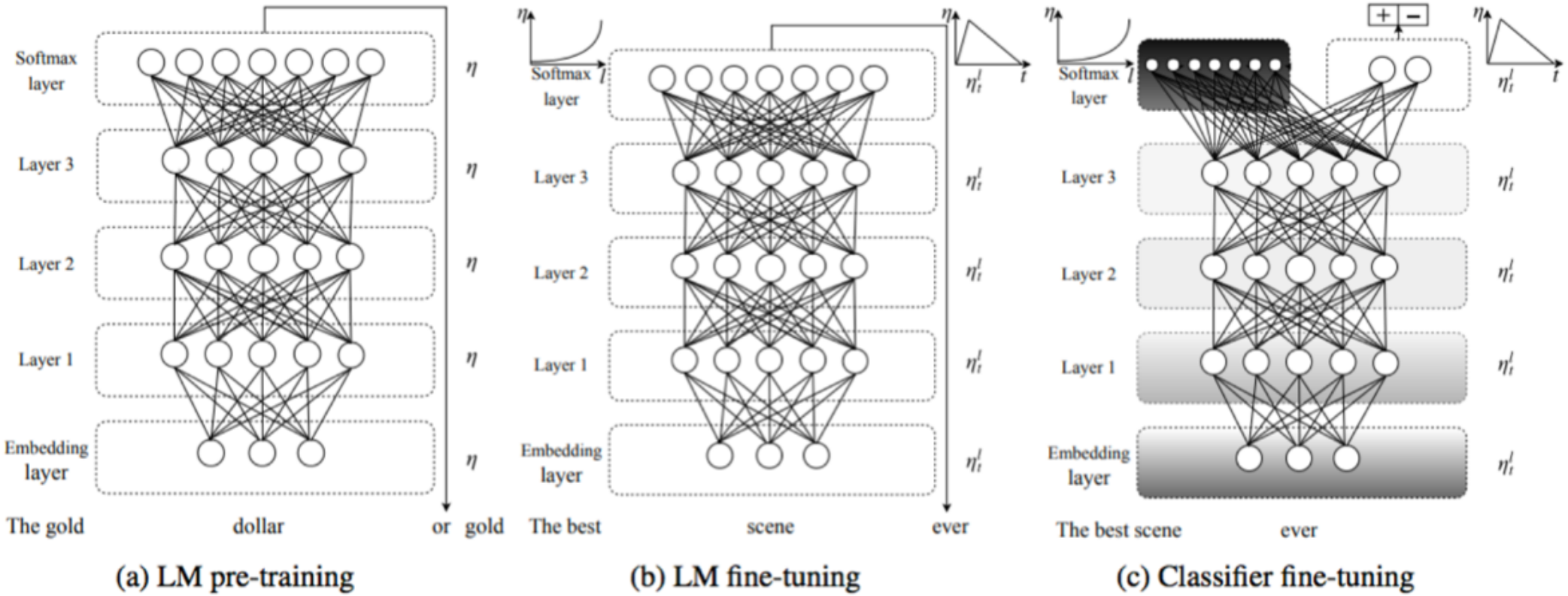
- Completely unsupervised
- Can leverage large unlabeled corpora (Wikipedia)

How? ULMFiT (1) -- pipeline

J. Howard, S. Ruder, "Universal Language Model Fine-tuning for Text Classification"

1. The LM is trained on a general-domain corpus to capture general features of the language in different layers (AWD-LSTM)
2. The full LM is fine-tuned on target task data using discriminative fine-tuning and slanted triangular learning rates (STLR)
3. The classifier is fine-tuned on the target task using gradual unfreezing

How? ULMFiT (2)



(a) LM pre-training (b) LM fine-tuning (c) Classifier fine-tuning

How? ULMFiT (3) – LM model

- A regular LSTM with various tuned dropout hyperparameters (AWD-LSTM)
  - Variational dropout
  - Embedding dropout
  - Etc

Model	Validation perplexity	Test perplexity	Paper / Source
AWD-LSTM-MoS + dynamic eval (Yang et al., 2018)*	42.41	40.68	Breaking the Softmax Bottleneck: A High-Rank RNN Language Model
AWD-LSTM + dynamic eval (Krause et al., 2017)*	46.4	44.3	Dynamic Evaluation of Neural Sequence Models
AWD-LSTM + continuous cache pointer (Merity et al., 2017)*	53.8	52.0	Regularizing and Optimizing LSTM Language Models
AWD-LSTM-MoS (Yang et al., 2018)	63.88	61.45	Breaking the Softmax Bottleneck: A High-Rank RNN Language Model
AWD-LSTM (Merity et al., 2017)	68.6	65.8	Regularizing and Optimizing LSTM Language Models

S. Merity, N. Keskar, R.Socher, Regularizing and Optimizing LSTM Language Models

How? ULMFiT (4) – discriminative fine-tuning

- As different layers capture different types of information (Yosinskiet al, 2014), they should be fine-tuned to different extents.

$$\theta_t^l = \theta_{t-1}^l - \eta^l \nabla_{\theta^l} J(\theta)$$

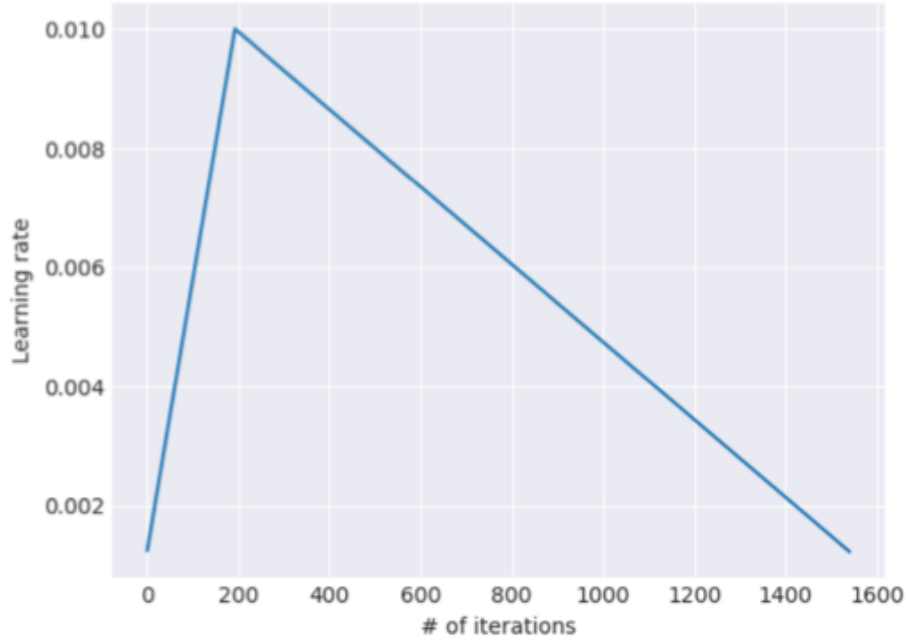
$$\eta^{l-1} = \eta^l / 2.6$$

How? ULMFiT (5) – STLR

$$cut = \lfloor T \cdot cut\_frac \rfloor$$

$$p = \begin{cases} t/cut, & \text{if } t < cut \\ 1 - \frac{t - cut}{cut \cdot (1/cut\_frac - 1)}, & \text{otherwise} \end{cases}$$

$$\eta_t = \eta_{max} \cdot \frac{1 + p \cdot (ratio - 1)}{ratio}$$



How? ULMFiT (6) – Concat pooling

- The signal in text classification tasks is often contained in a few words, which may occur anywhere in the document

$$\mathbf{H} = \{\mathbf{h}_1, \ldots, \mathbf{h}_T\}:$$

$$\mathbf{h}_c = [\mathbf{h}_T, \texttt{maxpool}(\mathbf{H}), \texttt{meanpool}(\mathbf{H})]$$

How? ULMFiT (7) – Gradual unfreezing

Rather than fine-tuning all layers at once, which risks **catastrophic forgetting**, they propose to **gradually unfreeze** the model starting from the last layer as this contains **the least general knowledge** (Yosinski et al, 2014)

1. Unfreeze i^{th} layer, $i = \{n, n-1, \dots, 1\}$
2. Train for one epoch
3. Repeat

How good? (1)

- Sentiment analysis (IMDB)
- Question Classification (TREC)
- Topic classification (AG news and DBpedia)

	Model	Test		Model	Test
IMDb	CoVe (McCann et al., 2017)	8.2	TREC-6	CoVe (McCann et al., 2017)	4.2
	oh-LSTM (Johnson and Zhang, 2016)	5.9		TBCNN (Mou et al., 2015)	4.0
	Virtual (Miyato et al., 2016)	5.9		LSTM-CNN (Zhou et al., 2016)	3.9
	ULMFiT (ours)	**4.6**		ULMFiT (ours)	**3.6**

Table 2: Test error rates (%) on two text classification datasets used by McCann et al. (2017).

	AG	DBpedia	Yelp-bi	Yelp-full
Char-level CNN (Zhang et al., 2015)	9.51	1.55	4.88	37.95
CNN (Johnson and Zhang, 2016)	6.57	0.84	2.90	32.39
DPCNN (Johnson and Zhang, 2017)	6.87	0.88	2.64	30.58
ULMFiT (ours)	**5.01**	**0.80**	**2.16**	**29.98**

Table 3: Test error rates (%) on text classification datasets used by Johnson and Zhang (2017).

How good? (2)

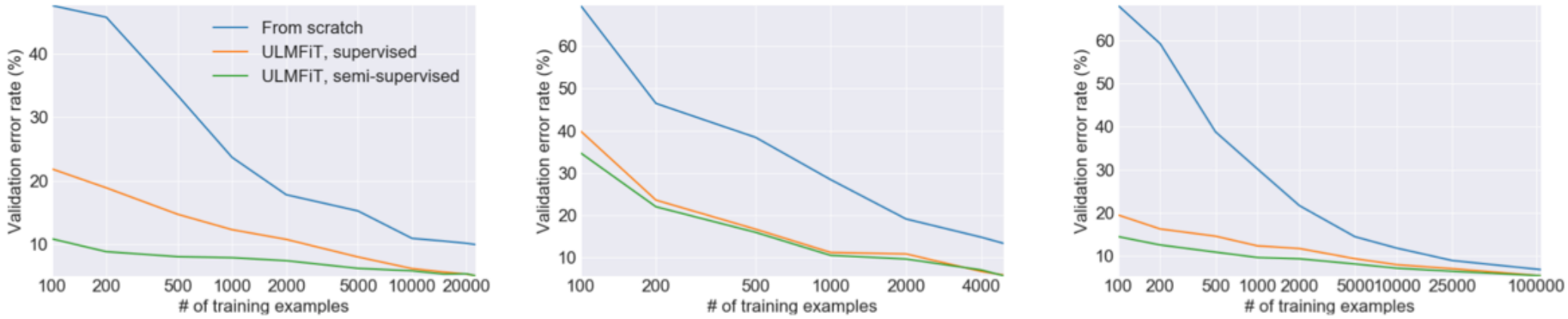


Figure 3: Validation error rates for supervised and semi-supervised ULMFiT vs. training from scratch with different numbers of training examples on IMDb, TREC-6, and AG (from left to right).

What's next?

- Pre-train your NLP models
- Use pre-trained ELMo (PyTorch, Tensorflow)
- Use tensorflow hub (universal sentence encoder, ELMo)
- Use fast.ai library and pre-trained ULMFiT models
- Read papers
  - Improving Language Understanding by Generative Pre-Training (OpenAI, 2018)
  - ULMFiT: Universal Language Model Fine-tuning for Text Classification (Fast AI, 2018)
  - ELMo: Deep Contextualized Word Vectors (AllenAI, 2018)