A Gentle Introduction to Deep Nets and Opportunities for the Future

Kenneth Church
Baidu, Sunnyvale, USA
Kenneth.Ward.Church@gmail.com

Valia Kordoni
Humboldt-Universitaet zu Berlin, Germany
evangelia.kordoni@anglistik.hu-berlin.de

Gary Marcus
NYU & Robust.AI
gary.marcus@icloud.com

Ernest Davis
New York University
davise@cims.nyu.edu

Yanjun Ma & Zeyu Chen
Baidu, Beijing, China
mayanjun02@baidu.com

Abstract

The first half of this tutorial will make deep nets more accessible to a broader audience, following "Deep Nets for Poets" and "A Gentle Introduction to Fine-Tuning." We will also introduce, *gft* (general fine tuning), a little language for fine tuning deep nets with short (one line) programs that are as easy to code as regression in statistics packages such as R using glm (general linear models). Based on the success of these methods on a number of benchmarks, one might come away with the impression that deep nets are all we need. However, we believe the glass is half-full: while there is much that can be done with deep nets, there is always more to do. The second half of this tutorial will discuss some of these opportunities.

1 Introduction

This tutorial is split into two parts:

A Glass is half-full: deep nets can do much
B Glass is half-empty: there is always more to do

Part A will make deep nets more accessible to a broader audience (Church et al., 2021b,a) by introducing *gft* (General Fine-Tuning), a new "little language"[1] for deep nets that is similar to *glm* (general linear models) in the statistics package R.[2] *gft* code will be posted on the tutorial website.[3]

2 Part A: Glass is Half-Full

2.1 The Standard Recipe

Following (Devlin et al., 2019; Howard and Ruder, 2018), it has become standard practice to use the 3-step recipe in Table 1, with an emphasis on pre-trained (foundation/base) models (Bommasani et al., 2021). *gft* prefers the terms, *fit* and *predict*, which have a long tradition in statistics, and predate relatively recent work on deep nets.

Step	*gft*	Standard Terminology
1		Pre-Training
2	fit	Fine-Tuning
3	predict	Inference

Table 1: 3-Step recipe has become standard practice

gft makes it easy to use models and datasets on hubs: HuggingFace[4] and PaddleHub/PaddleNLP.[5] The hubs are large (30k models and 3k datasets), and growing quickly (3x/year). The challenge is to make these amazing resources more accessible to a diverse user-base. One does not need to know python and machine learning to use an off-the-shelf regression package. So too, deep nets should not require much (if any) programming skills.

2.2 Examples of Fit (aka Fine-Tuning)

Fit takes a pre-trained model, f_{pre} (BERT), and uses a dataset (emotion) to output a post-trained model, f_{post} (to $outdir):

```
gft_fit --data "H:emotion" \
    --model "H:bert-base-cased" \
    --eqn "classify:label~text" \
    --output_dir "$outdir"
```

Listing 1: Example of *gft_fit*

The next example is similar but uses a model and a dataset from PaddleNLP. *gft* supports mixing and matching models and datasets from different hubs.

```
gft_fit --data "P:chnsenticorp" \
    --model "P:ernie-tiny" \
    --eqn "classify:label~text" \
    --output_dir "$outdir"
```

Listing 2: H and P refer to HuggingFace and PaddleNLP

[1] Little languages were advocated by Bentley (1986) and the Unix group. Little languages such as AWK (Aho et al., 1987) make it easy to solve remarkably powerful tasks with short (often one-line) programs.

[2] https://www.r-project.org/

[3] https://github.com/kwchurch/ACL2022_deepnets_tutorial

[4] https://huggingface.co/

[5] https://github.com/PaddlePaddle

Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts, pages 1 - 6

–data arg	–eqn arg
H:glue,cola	classify: label ∼ sentence
H:glue,sst2	classify: label ∼ sentence
H:glue,wnli	classify: label ∼sentence
H:glue,mrpc	classify: label ∼ sentence1 + sentence2
H:glue,rte	classify: label ∼sentence1 + sentence2
H:glue,qnli	classify: label ∼ question + sentence
H:glue,qqp	classify: label ∼question1 + question2
H:glue,sstb	regress: label ∼sentence1 + sentence2
H:glue,mnli	classify: label ∼ premise + hypothesis

Table 2: *gft* solutions for GLUE (Wang et al., 2018)

–data arg	–eqn arg
squad	classify_spans: answers ∼ question + context
tweet_eval,hate	classify: label ∼ text
conll2003	classify_tokens: pos_tags ∼ tokens
conll2003	classify_tokens: ner_tags ∼ tokens
conll2003	classify_tokens: chunk_tags ∼ tokens
timit_asr	ctc: text ∼ audio

Table 3: *gft* solutions for more benchmarks

Short (1-line) gft programs can fit (fine-tune) many benchmarks, as illustrated in Tables 2-3.

2.3 gft Cheatsheet

gft supports the following functions:

1. *fit* (*aka* fine-tuning): $f_{pre} + data \rightarrow f_{post}$
2. *predict* (*aka* inference): $f(x) = \hat{y}$, where x is an input from a dataset or from *stdin*
3. *eval*: $f + data \rightarrow score$
4. *summary*: search hubs for popular datasets, models and tasks, and provide snippets.
5. *cat_data*: output dataset on *stdout*

There are four major arguments:

1. –data: a dataset on a hub, or a local file
2. –model: a model on a hub, or a local file
3. –task: e.g., classify, regress[6]
4. –eqn (e.g., classify:$y \sim x_1 + x_2$), where a task appears before the colon, and variables refer to columns in the dataset.

The *gft* interpreter is based on examples from hubs.[7] [8] Hubs encourage users to modify 500+ lines of pytorch as necessary if they want to change models, datasets and/or tasks. *gft* generalizes the examples so users can do much of that in a single line of *gft* code (with comparable performance).[9]

2.4 Some Simple Examples

2.4.1 Search

As mentioned above, users are overwhelmed with an embarrassment of riches. How do we find the good stuff on the hubs? The following outputs snippets for datasets, models and tasks:

```
m=bhadresh-savani/roberta-base-emotion
gft_summary --data "H:emotion"
gft_summary --model "H:$m"
gft_summary --task "H:classify"
```

Listing 3: Models/datasets/tasks → snippets

Search for datasets and models that contain the substring: *emotion*, sorted by downloads:

```
query=H:__contains__emotion
gft_summary --data "$query" --topn 5
gft_summary --model "$query" --topn 5
```

Listing 4: Searching for best *emotion* models/datasets

To find the most downloaded datasets and models, set the query to the empty string:

```
query=H:__contains__
gft_summary --data "$query" --topn 5
gft_summary --model "$query" --topn 5
```

Listing 5: Searching for best of everything

2.4.2 Predict (*aka* Inference)

After having found the *good* stuff, how do we use it? *gft_predict* takes input, x, from stdin and outputs predictions, $\hat{y}$.

```
c=H:classify
tc=H:token-classification
# sentiment classification
echo "I love you"|gft_predict --task $c
# emotion classification
echo "I love you"|
  gft_predict --task $c --model $m
# NER (Named Entity Recognition)
echo "I love New York"|
  gft_predict --task $tc
```

[6] Currently supported tasks are: classify (*aka* text-classification), classify_tokens (*aka* token-classification), classify_spans (*aka* QA, question-answering), classify_images (*aka* image-classification), classify_audio (*aka* audio-classification), regress, text-generation, MT (*aka* translation), ASR (*aka* ctc, automatic-speech-recognition), fill-mask. Tasks in parentheses are aliases.

[7] https://github.com/huggingface/transformers/blob/master/examples/pytorch/

[8] https://github.com/PaddlePaddle/PaddleNLP/tree/develop/examples

[9] *gft* supports most of the arguments in the examples on the hubs, so it is possible to tune hyperparameters such as batch size, learning rate and stopping rules. Tuning is important for SOTA-chasing (Church and Kordoni, 2022), though default settings are recommended for most users who prefer results that are easy to replicate, and reasonably competitive.

```
# cloze task (fill in the <mask>)
echo "I <mask> you"|
  gft_predict --task H:fill-mask
```

Listing 6: Examples of *gft_predict*

gft_predict can also input from a dataset split, and outputs a prediction, $\hat{y}$, for each x in the split:

```
eqn="classify:label~text"
gft_predict --eqn "$eqn" --model $m \
   --data H:emotion --split test
```

Listing 7: Input from a dataset (instead of *stdin*)

2.4.3 Evaluation

If we replace *gft_predict* (above) with *gft_eval* (below), then we obtain a single score (instead of a $\hat{y}$ for each x):

```
gft_eval --eqn "$eqn" --model $m \
   --data H:emotion --split test
```

Listing 8: Evaluating a model on a dataset

2.4.4 Ease of Use, Popularity & SOTA

Given an embarrassment of riches, how do we choose the best model? The literature emphasizes SOTA (state-of-the-art), hubs reward downloads, and *gft* advocates ease-of-use.

Table 4 reports accuracy for a few models containing "MRPC,"[10] as well as two custom models. *gft* makes it easy to achieve competitive results, close to distilbert (compressed) models. One can outperform models on the hubs, by tuning hyperparameters as Yuchen Bian did. Tuning is possible in *gft* (but not recommended), as discussed in footnote 9. The validation accuracy in Table 4 are well below test accuracy in Table 5,[11] [12] suggesting that popular/easy-to-use/compressed models are well below SOTA (though we should not compare validation accuracy with test accuracy).

2.5 Conclusions to Part A

Higher level (little) languages like *gft* have many advantages over examples found on hubs: short (1-line) programs are easier to read and write, more transparent and more portable (across hubs). *gft* code and hundreds of examples can be found on the tutorial website (see footnote 3).

[10] We tested 22 models from HuggingFace and 135 models from Yuchen Bian (personal communication). To save space, results are reported for the best of Bian's models, the top 3 HuggingFace models, and models with 100+ downloads.

[11] https://paperswithcode.com/sota/semantic-textual-similarity-on-mrpc

[12] https://gluebenchmark.com/leaderboard

Model	VAcc	D
C:RoBERTa large, tuned by Yuchen Bian	0.924	
H:textattack/roberta-base-MRPC	0.912	1623
H:textattack/albert-base-v2-MRPC	0.897	175
H:mrm8488/deberta-v3-small-finetuned-mrpc	0.892	30
H:textattack/bert-base-uncased-MRPC	0.877	10,133
H:textattack/distilbert-base-uncased-MRPC	0.858	108
H:ajrae/bert-base-uncased-finetuned-mrpc	0.858	115
C:*gft_fit* example (BERT with no tuning)	0.853	
H:textattack/distilbert-base-cased-MRPC	0.784	122

Table 4: *gft* achieves VAcc (accuracy on validation split) close to distilbert (compressed) models. HuggingFace models were selected using *gft_summary* to find popular models by downloads (D).

Source	Test Accuracy
GLUE Leaderboard (L)	0.945
Papers with code (PWC)	0.937
Human Baseline (HB)	0.863

Table 5: SOTA (state-of-the-art) for MRPC (GLUE). See footnote 11 for PWC, and 12 for L & HB.

The point of Part A is to demystify deep nets. No one would suggest that regression-like methods are magical, or even artificially intelligent.

The point of Part B is to set appropriate expectations. There are many classic problems in knowledge representation, cognitive science and linguistics that go beyond regression-like methods discussed in Part A.

3 Part B: Opportunities for Improvement

Language models (LMs) are based on (Firth, 1957): "You shall know a word by the company it keeps" and Zellig Harris's (1954) "distributional hypothesis." By construction, this approach learns many aspects of language, some more desirable (fluency, collocations, word patterns) and some less desirable (*biases* (Bender et al., 2021)). However, there are many aspects that are not learned: *truth* (logical form, temporal/spatial logic and possible worlds), *meaning*, *purpose* (planning (Kautz et al., 1986; Litman and Allen, 1987), discourse structure) and *commonsense knowledge* (time and space). These topics have been studied for decades in AI and knowledge representation and for centuries in linguistics and philosophy.

3.1 Truth

> To the extent that a use case places importance on the truth of the outputs provided, it is not a good fit for GPT-3 (Dale, 2021)

LMs have a tendency to "hallucinate" when summarizing documents. The output sounds plausible, but may add embellishments to the input. More generally, LMs tend to make up "alternative facts" faster than they can be fact-checked. This may well be their most dangerous failing; people might believe some of these conspiracy theories.

3.2 Meaning

A vivid example of challenges with meaning is Ettinger's (2020) study of negation. If you ask BERT to fill in the blank in:

- A robin is a ____ .
- A robin is not a ____.

the top answer is: "bird," in both cases. There are few wrong answers in the second case, but "bird" is one of them.

3.3 Purpose, Planning & Document Structure

LMs generate text word-by-word without looking ahead and thinking about the larger picture. Short outputs are remarkably fluent, but longer outputs tend to meander aimlessly. Dialogue systems optimize for smoothness from the most recent turn. Such short-term thinking may not be helpful to the user (Grice, 1975). In one notorious case, a GPT-3 chatbot in the medical domain advised a patient to commit suicide (Rousseau and Baudelaire, 2020). More generally, LMs produce non-sequiturs, contradictions, tautologies, echolalia (Metz, 2020).

3.4 Commonsense knowledge

Commonsense knowledge is basic knowledge of how the world works (Davis and Marcus, 2015). We tested GPT-3's command of spatial and temporal knowledge with questions such as:

Time: Who came first, Thomas Jefferson or John F. Kennedy?

Space: Which is further from Liverpool, England: Brussels, Belgium or Portland, Oregon?

GPT-3 performed at chance on space, and only slightly better on time. LMs can output dates for historical figures and coordinates of cities, if asked directly, but LMs struggle to use this knowledge for questions such as the ones above.

The questions in our experiment involve particularly simple forms of temporal and spatial reasoning. Many texts make use of complex temporal relations such as possible worlds[13] and hypothetical events (such as planning, hoping, fearing, and preventing) (Gordon and Hobbs, 2017). Text often make use of complex features involving shapes and spatial relations (Davis, 2013).

Time[14] and space (Bloom, 1999) have been extensively studied in linguistics and philosophy. It is natural to model time based on tense. One approach,[15] starts with speech time, S, reference time, R, and event time, E.[16]

past perfect (*had slept*) $E < R < S$
simple past (*slept*) $E \approx R, E < S, R < S$

There are also natural connections between linguistic constructions such as subjunctive (*would, could, should*) and possible worlds. More generally, much of the work in linguistics assumes a rich set of connections between surface representations (syntax) and deeper structures (semantics/pragmatics).

4 Conclusions: Some Paths Forward

Some of these opportunities can be addressed by relatively easy patches to Firth-based methods. For example, biases can be mitigated in the short term by vetting the training corpus (Hovy and Prabhumoye, 2021). Similarly, penalty terms can be added to the objective function to discourage hallucinations (Durmus et al., 2020). Fine-tuning on a corpus of commonsense knowledge can help with violations of commonsense (Zhang et al., 2021).

In the long term, it may be helpful to consider more radical alternatives (Marcus and Davis, 2019). Part A described some recent advances that have been remarkably successful, though to make long term advances beyond that, it may be necessary to take advantage of more diverse interdisciplinary approaches that include Firth-based methods, as well as decades of work on Knowledge Representation in AI, and centuries of work in linguistics and philosophy.

[13] https://plato.stanford.edu/entries/possible-worlds/

[14] https://plato.stanford.edu/entries/logic-temporal/

[15] https://plato.stanford.edu/entries/reichenbach/#AxiTheRel192

[16] In the past perfect, event time precedes reference time, which precedes speech time. In contrast, in the simple past, event time coincides with reference time, while both precede speech time.

References

Alfred V Aho, Brian W Kernighan, and Peter J Weinberger. 1987. *The AWK programming language*. Addison-Wesley Longman Publishing Co., Inc.

Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pages 610–623.

Jon Louis Bentley. 1986. Little languages. *Commun. ACM*, 29(8):711–721.

Paul Bloom. 1999. *Language and space*. MIT press.

Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri Chatterji, Annie Chen, Kathleen Creel, Jared Quincy Davis, Dora Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren Gillespie, Karan Goel, Noah Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab, Pang Wei Kohd, Mark Krass, Ranjay Krishna, Rohith Kuditipudi, Ananya Kumar, Faisal Ladhak, Mina Lee, Tony Lee, Jure Leskovec, Isabelle Levent, Xiang Lisa Li, Xuechen Li, Tengyu Ma, Ali Malik, Christopher D. Manning, Suvir Mirchandani, Eric Mitchell, Zanele Munyikwa, Suraj Nair, Avanika Narayan, Deepak Narayanan, Ben Newman, Allen Nie, Juan Carlos Niebles, Hamed Nilforoshan, Julian Nyarko, Giray Ogut, Laurel Orr, Isabel Papadimitriou, Joon Sung Park, Chris Piech, Eva Portelance, Christopher Potts, Aditi Raghunathan, Rob Reich, Hongyu Ren, Frieda Rong, Yusuf Roohani, Camilo Ruiz, Jack Ryan, Christopher Ré, Dorsa Sadigh, Shiori Sagawa, Keshav Santhanam, Andy Shih, Krishnan Srinivasan, Alex Tamkin, Rohan Taori, Armin W. Thomas, Florian Tramèr, Rose E. Wang, William Wang, Bohan Wu, Jiajun Wu, Yuhuai Wu, Sang Michael Xie, Michihiro Yasunaga, Jiaxuan You, Matei Zaharia, Michael Zhang, Tianyi Zhang, Xikun Zhang, Yuhui Zhang, Lucia Zheng, Kaitlyn Zhou, and Percy Liang. 2021. On the opportunities and risks of foundation models.

Kenneth Ward Church, Zeyu Chen, and Yanjun Ma. 2021a. Emerging trends: A gentle introduction to fine-tuning. *Natural Language Engineering*, 27(6):763–778.

Kenneth Ward Church and Valia Kordoni. 2022. Emerging trends: Sota-chasing. *Natural Language Engineering*, 28(2):249–269.

Kenneth Ward Church, Xiaopeng Yuan, Sheng Guo, Zewu Wu, Yehua Yang, and Zeyu Chen. 2021b. Emerging trends: Deep nets for poets. *Natural Language Engineering*, 27(5):631–645.

Robert Dale. 2021. Gpt-3: What's it good for? *Natural Language Engineering*, 27(1):113–118.

Ernest Davis. 2013. Qualitative spatial reasoning in interpreting text and narrative. *Spatial Cognition & Computation*, 13(4):264–294.

Ernest Davis and Gary Marcus. 2015. Commonsense reasoning and commonsense knowledge in artificial intelligence. *Communications of the ACM*, 58(9):92–103.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Esin Durmus, He He, and Mona Diab. 2020. FEQA: A question answering evaluation framework for faithfulness assessment in abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5055–5070, Online. Association for Computational Linguistics.

Allyson Ettinger. 2020. What BERT is not: Lessons from a new suite of psycholinguistic diagnostics for language models. *Transactions of the Association for Computational Linguistics*, 8:34–48.

John R Firth. 1957. A synopsis of linguistic theory, 1930-1955. *Studies in linguistic analysis*.

Andrew S Gordon and Jerry R Hobbs. 2017. *A formal theory of commonsense psychology: How people think people think*. Cambridge University Press.

Herbert P Grice. 1975. Logic and conversation. In P. Cole and J. Morgan, editors, *Syntax and Semantices: Vol 3: Speech Acts*. Academic Press, London.

Zellig S Harris. 1954. Distributional structure. *Word*, 10(2-3):146–162.

Dirk Hovy and Shrimai Prabhumoye. 2021. Five sources of bias in natural language processing. *Language and Linguistics Compass*, 15(8):e12432.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia. Association for Computational Linguistics.

Henry A Kautz, James F Allen, et al. 1986. Generalized plan recognition. In *AAAI*, volume 86, page 5.

Diane J Litman and James F Allen. 1987. A plan recognition model for subdialogues in conversations. *Cognitive science*, 11(2):163–200.

Gary Marcus and Ernest Davis. 2019. *Rebooting AI: Building artificial intelligence we can trust*. Pantheon Press.

Cade Metz. 2020. When A.I. falls in love. *The New York Times*. Nov. 24, 2020.

Anne-Laure Rousseau and Clément Baudelaire. 2020. Doctor GPT-3: Hype or reality? *Nabla*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Zhihan Zhang, Xiubo Geng, Tao Qin, Yunfang Wu, and Daxin Jiang. 2021. Knowledge-aware procedural text understanding with multi-stage training. In *Proceedings of the Web Conference 2021*, pages 3512–3523.