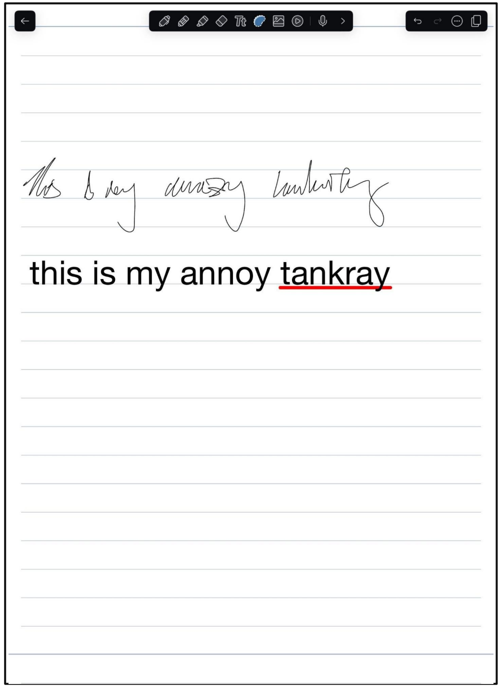
Recognizing Illegible Handwriting

By Khoi Nguyen, Alex Clinton,
Zhuoming Liu, and Thomas Zeng

1.1 Problem motivation

- Thomas likes to take notes on his iPad using an app called notability
- This app has a optical character recognition ***(OCR)*** feature which recognizes your handwriting so that you can filter and search through your notes
- However, Thomas has ***bad handwriting***...very bad handwriting
- Inhibits the the app ability to index his notes and to let him quickly locate him
- Goal: Create a better OCR tool so that Thomas (and others with messy handwriting) can make better use of their digital notes



1.2 How would you classify this sentence?

- Google Lens: Sor resemmes that probre and sell homed.
- Microsoft CoPilot: So it seems that people can still read.
- Actual Text: For restaurants that produce and sell bread.

1.3 Problem statement

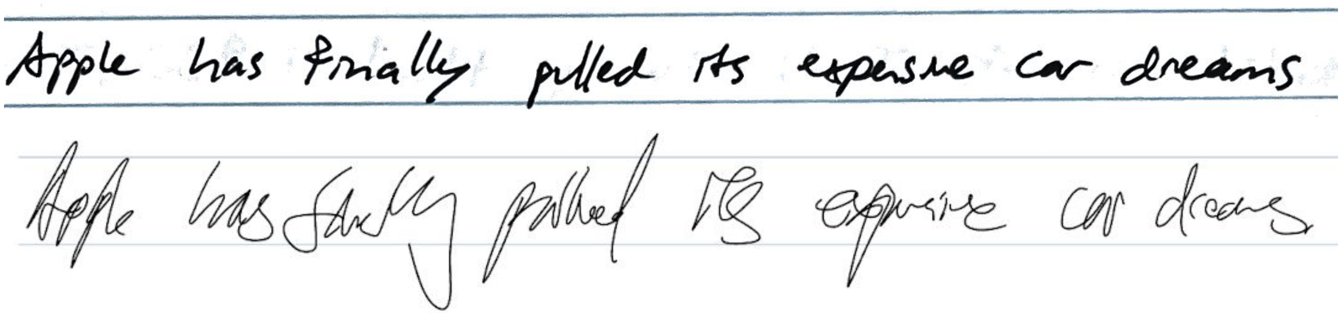
- Typically, handwriting recognition is considered to be a solved problem
- However we argue that our problem is more difficult because it is not "***well defined***", in the sense that it is hard for humans to recognize what Thomas writes
- In general, we wish to efficiently create a ***personalized*** handwriting recognition model, even when their writing is ***beyond the recognition capabilities of a human***

2 Contents

1. Dataset preparation
2. Evaluation Metric
3. Naive fine tuning
4. Supervised domain adaptation
5. Transfer learning
6. Dual-decoder
7. Meta-learning

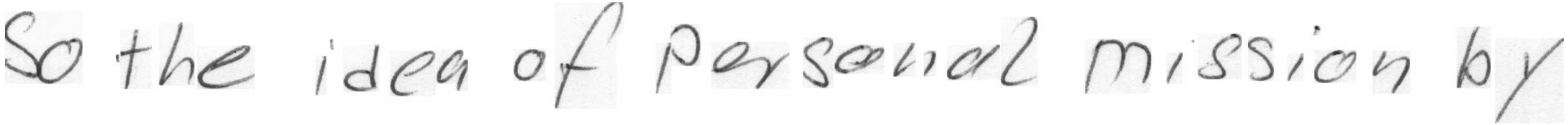
2.1.1 Dataset Preparation: custom datasets

- We create two datasets of 60 images each, one of Thomas' handwriting and one of Alex's handwriting
- Results in this talk uses 50 for train, 10 for test
- Data Augmentation
    - +/- 10 degree rotations
    - Randomly apply Gaussian smoothing



2.1.2 Dataset Preparation: IAM dataset

- Some methods that we use require data from many different writers
- For this purpose we use the IAM dataset which is comprised of
  - 13,353 images of handwritten lines of text
  - By 657 writers.
- This dataset has been widely used across many NLP tasks



2.2 Evaluation Metric

- Character error rate ***(CER)*:** Is a common metric in natural language processing tasks
- *CER* = (S + D + I) / N = (S + D + I) / (S + D + C)
  - S = substitutions
  - I = insertions
  - D = deletions
  - C = correct characters
  - N = total characters

- Note that CER is not always in the range [0,1] when I is large

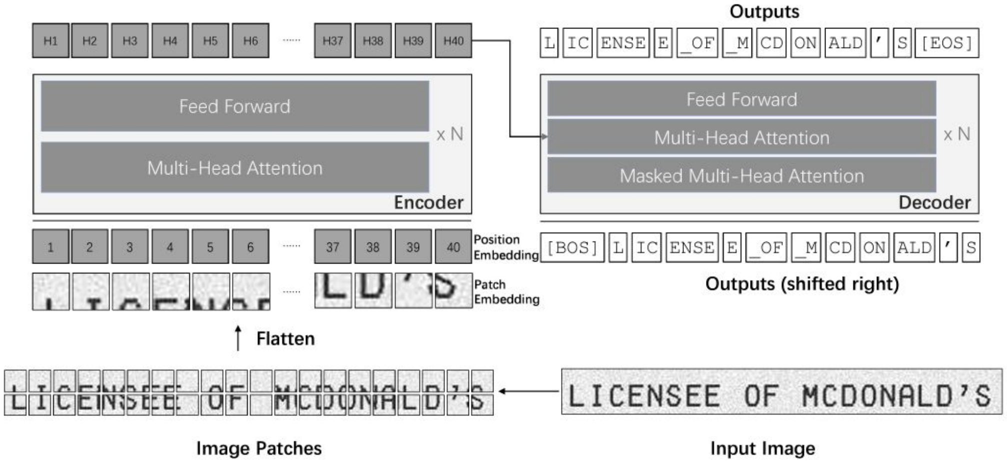
computer vision ⟶ Compute visionn

CER = 0.2

2.3 Naive fine tuning: method (Approach 1)

Method:

- TrOCR is a transformer based OCR model (already trained on IAM)
- We finetune either the whole model or just the decoder of TrOCR



Minghao Li et al. “Trocr: Transformer-based optical character recognition with pre-trained models”. In: Proceedings of the AAAI Conference on Artificial Intelligence . Vol. 37. 11. 2023, pp. 13094–13102.

2.3 Naive fine tuning: result

Results on Thomas and Alex datasets:

Finetune on	Dataset	CER
None	Thomas	5.06
Whole model	Thomas	0.84
Decoder only	Thomas	**0.69**
None	Alex	3.10
Whole model	Alex	0.79
Decoder only	Alex	**0.34**

Exploration on how many images we need:

# of Images	CER
50	0.65
20	0.89
10	0.92
5	1.74
1	2.62

Conclusion:

1. Fine Tuning on decoder has better performance.
2. Model performance not saturated even we use 50 image for fine tuning

2.4 Supervised domain adaptation (Approach 2)

Motivation:
Aim to bridge the statistical difference between the new dataset and the old dataset.

Common solution:
Retraining the normalization layer

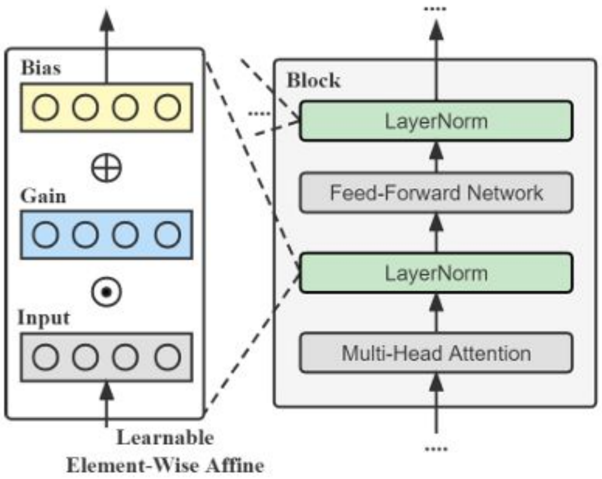


Figure 1: Illustration of our proposed LN-tuning.

Qi, Wang, et al. "Parameter-efficient tuning on layer normalization for pre-trained language models." *arXiv preprint arXiv:2211.08682* (2022).

2.4 Supervised domain adaptation: results

Results:

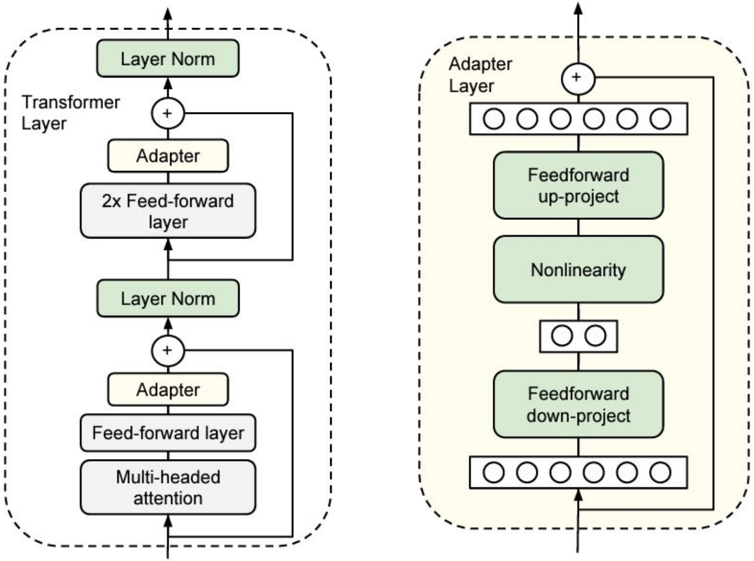
Retrain	Dataset	CER
None	Thomas	5.06
Decoder Norm layer	Thomas	3.99
All Norm layer	Thomas	**3.02**
None	Alex	3.10
Decoder Norm layer	Alex	2.44
All Norm layer	Alex	**1.20**

Conclusion: Retraining the Normalization can bridge the domain gap and improve the model performance.

2.5 Transfer learning (Approach 3)

Motivation:
Compared with OCR for printed letter and OCR for the messy handwriting can be regarded as the new task.

Proposed method:
Train an Adaptor in every transformer layer.



Houlsby, Neil, et al. "Parameter-efficient transfer learning for NLP." *International conference on machine learning*. PMLR, 2019.

2.5 Transfer learning: results

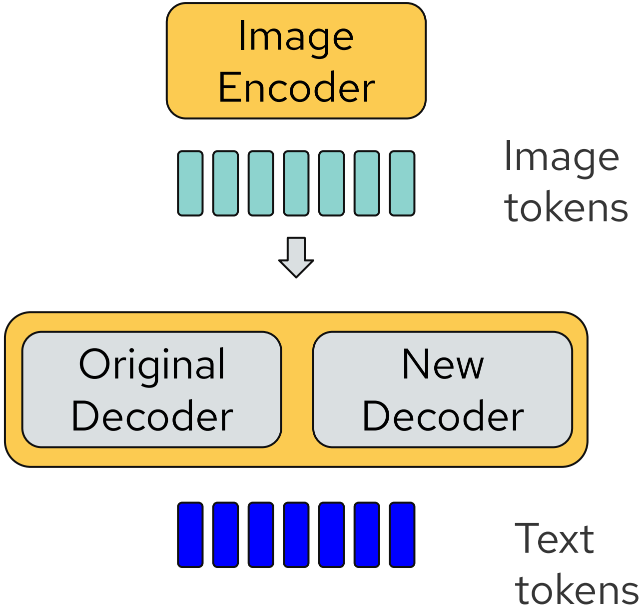
Method	Dataset	CER
None	Thomas	5.06
Adaptor	Thomas	**0.72**
None	Alex	3.10
Adaptor	Alex	**0.52**

Conclusion: Adaptor improves the model performance and also improves the training efficiency.

2.6 Dual-decoder (Approach 4)

Motivation:
Keep the original decoder to maintain the prior knowledge of the sentence structure. Train a new decoder learns the knowledge of the new task.

Proposed method:
Train an additional decoder.

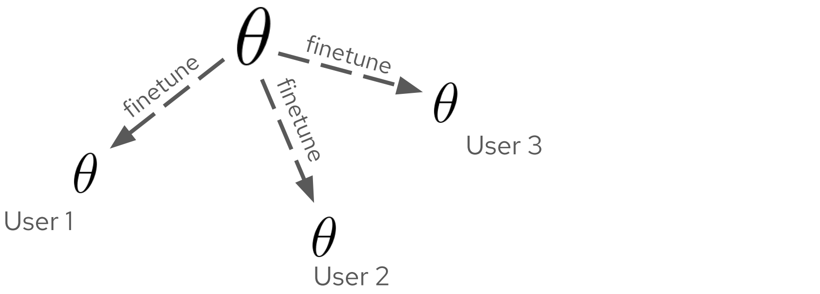


2.6 Dual-decoder: results

Retrain	Dataset	CER
None	Thomas	5.06
Dual decoder	Thomas	**0.62**
None	Alex	3.10
Dual decoder	Alex	**0.71**

Conclusion: The dual decoder improves the model performance by making use of the prior knowledge.

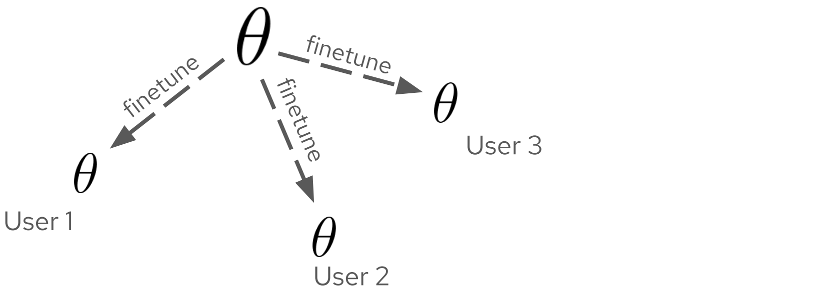
2.7 Meta learning (Approach 5)



Finn et al. 2017. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks.

2.7 Meta learning (Approach 5)

Training

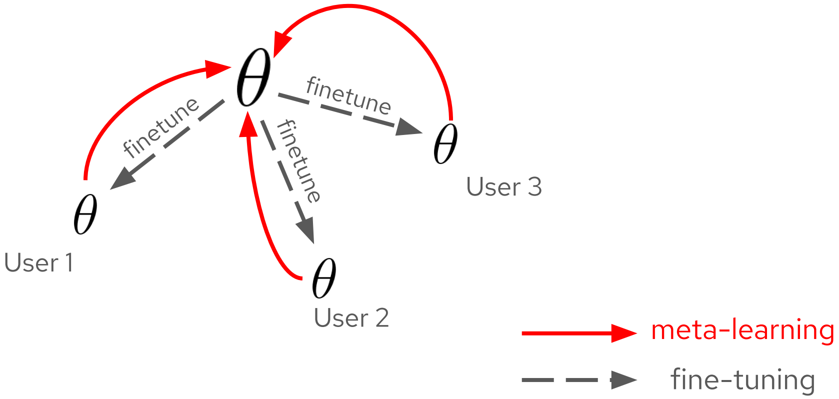


User 1, User 2, User 3 ~ IAM Dataset

Finn et al. 2017. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks.

2.7 Meta learning (Approach 5)

Training

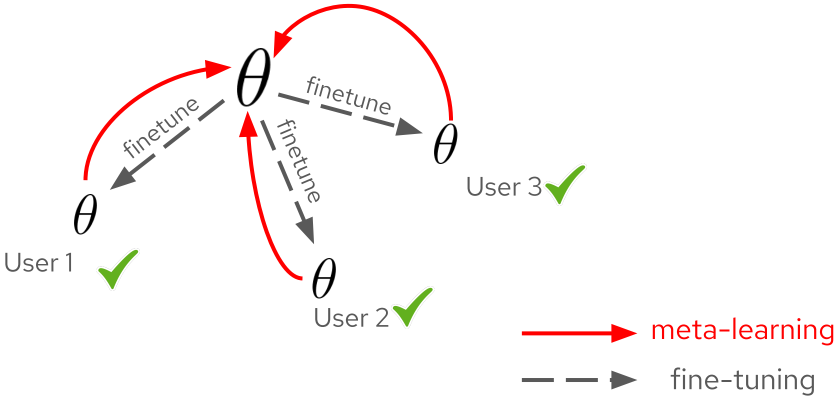


User 1, User 2, User 3 ~ IAM Dataset

Finn et al. 2017. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks.

2.7 Meta learning (Approach 5)

Training

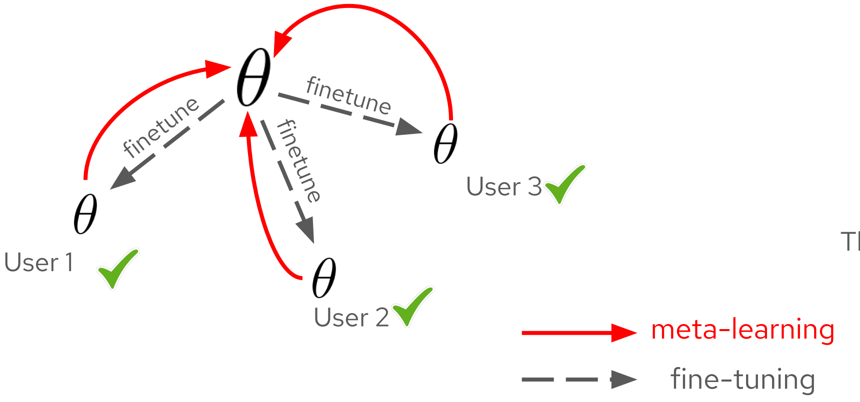


User 1, User 2, User 3 ~ IAM Dataset

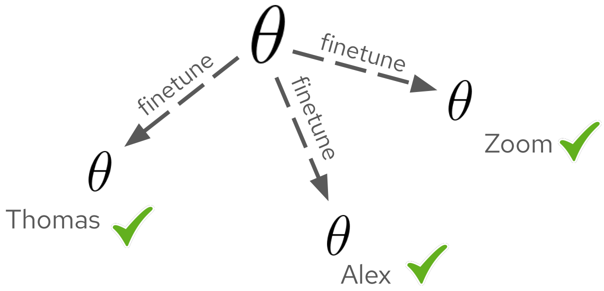
Finn et al. 2017. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks.

2.7 Meta learning (Approach 5)

Training



Testing



User 1, User 2, User 3 ~ IAM Dataset

Finn et al. 2017. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks.

2.8 Meta learning: results

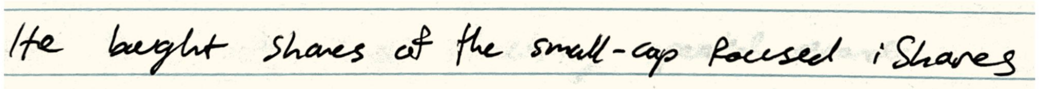
Checkpoint	Dataset	CER
Original checkpoint	Thomas	5.06
Original checkpoint (with finetuning)	Thomas	0.84
MAML checkpoint (with finetuning)	Thomas	0.78
Original checkpoint (with finetuning decoder)	Thomas	0.69
MAML checkpoint (with finetuning decoder)	Thomas	**0.51**
Original checkpoint	Alex	3.10
Original checkpoint (with finetuning)	Alex	0.79
MAML checkpoint (with finetuning)	Alex	0.77
Original checkpoint (with finetuning decoder)	Alex	0.34
MAML checkpoint (with finetuning decoder)	Alex	**0.26**

Conclusion: The MAML checkpoint can be used to improve finetuning efficiency.

2.9 Qualitative results


for restaurants that produce and sell bread

Predict: for restaurants that produce and sellbread
Ground-truth: for restaurants that produce and sell bread


He bought shares of the small-cap focused iShares

Predict: He bought shares of the small-cap
Ground-truth: He bought shares of the small-cap focused iShares

3 Future Works and Conclusion

- The MAML and dual decoder approaches yield the best performance

3 Future Works and Conclusion

- The MAML and dual decoder approaches yield the best performance
- Our final result of a CER of 0.26/0.51 (for Alex/Thomas) is still far from ideal (goal is ~0.02)

3 Future Works and Conclusion

- The MAML and dual decoder approaches yield the best performance
- Our final result of a CER of 0.26/0.51 (for Alex/Thomas) is still far from ideal (goal is ~0.02)

- Future works
  - Data Augmentation
  - Use more data
  - Use larger model

4 Additional experiments after presentation

- This experiment aims to explore the effect of adding additional training images, since the previous experiment shows that the model performance on Thomas dataset is still lag behind the model performance on Alex.
- We would like to see whether more training data can bridge the performance gap.

Number of Image	CER
0	5.06
60	0.51
100	0.216

- The experiments shows adding additional samples brings model performance on the Thomas dataset to a usable level close the gap between the alex and thomas.