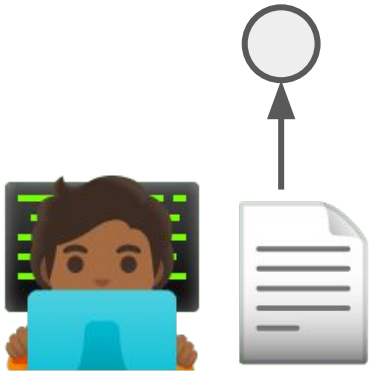
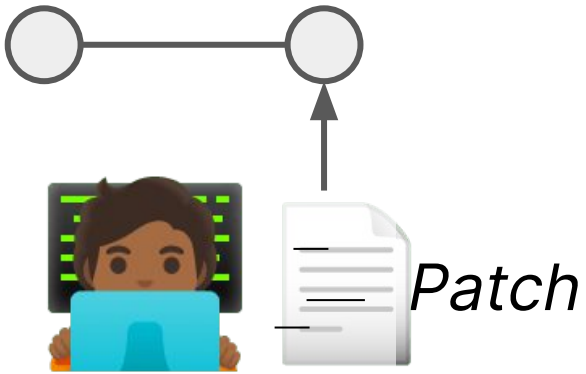
Infrastructure and Progress Towards the First Community-Built and Continually-Improved Model

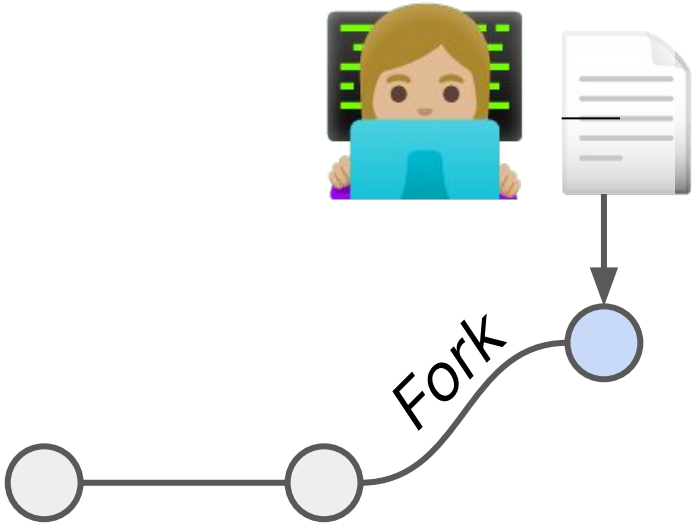
Colin Raffel

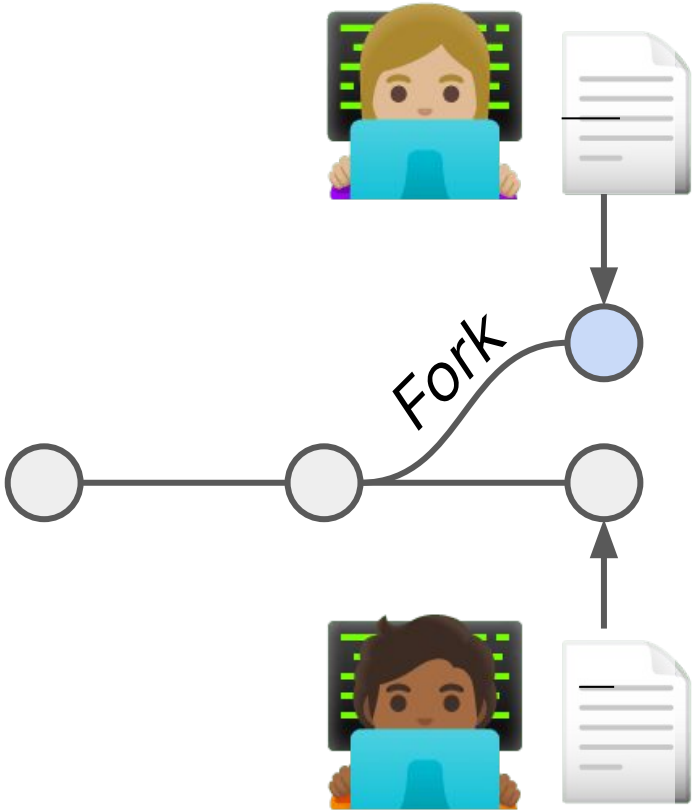
Collaborative, Communal, Continual {CCC} ML

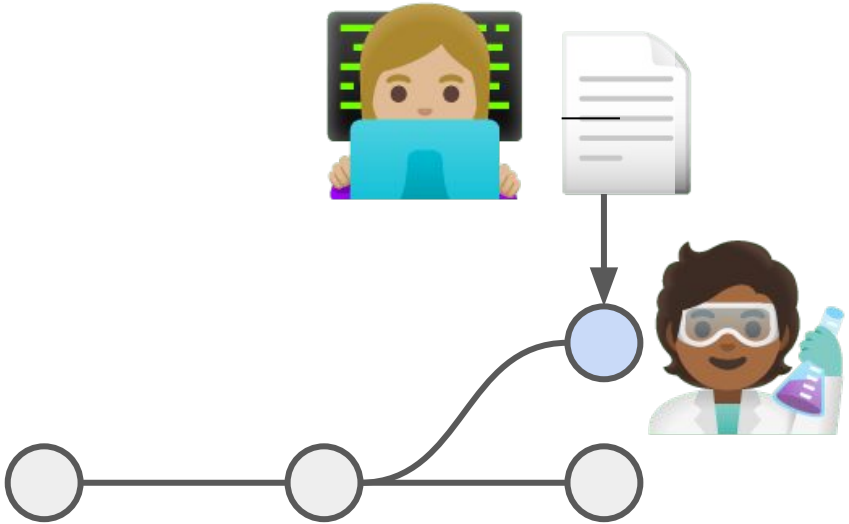
- Where are we in terms of making CCCML possible?
- We can build a useful VCS for CCCML now.
- We’re building a `git`-based VCS for CCCML.
- We’re building a CCC model to test it out.

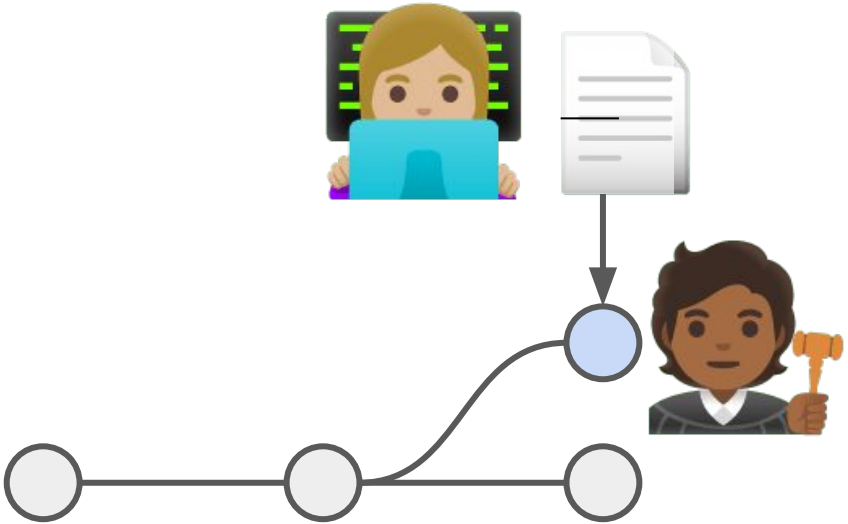


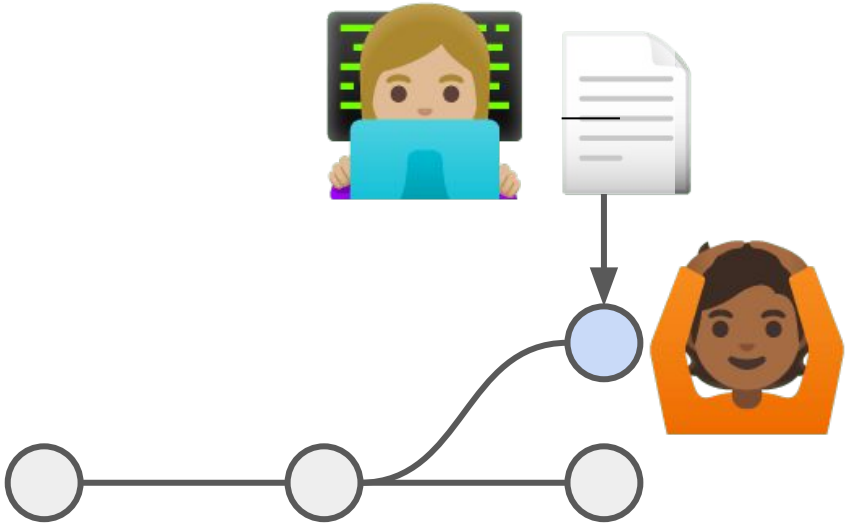

Patch

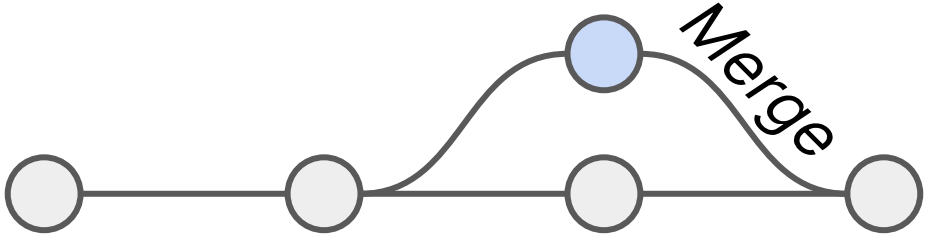

Fork

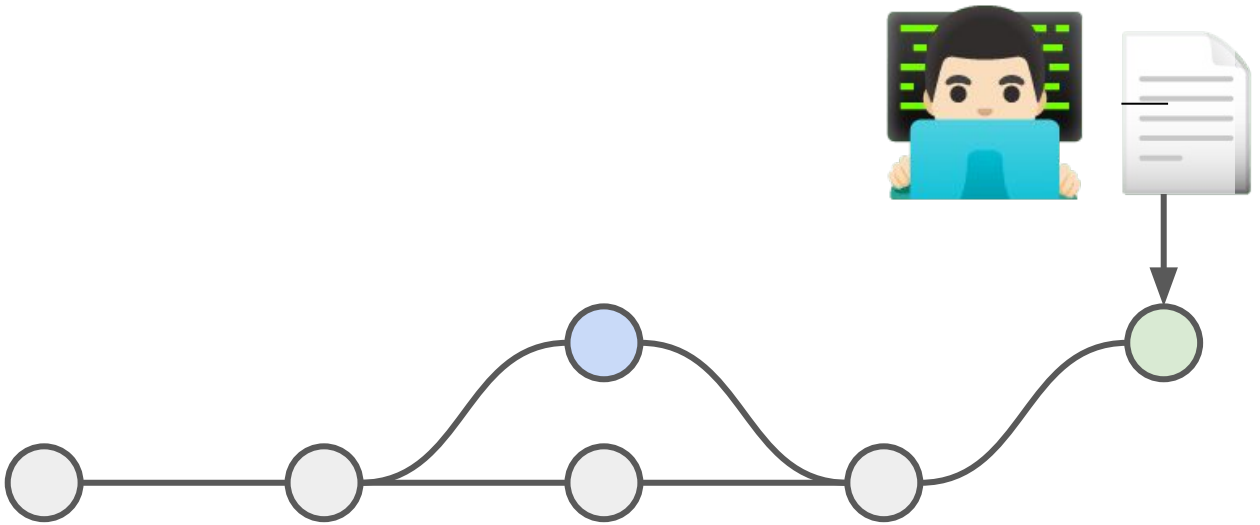

Fork

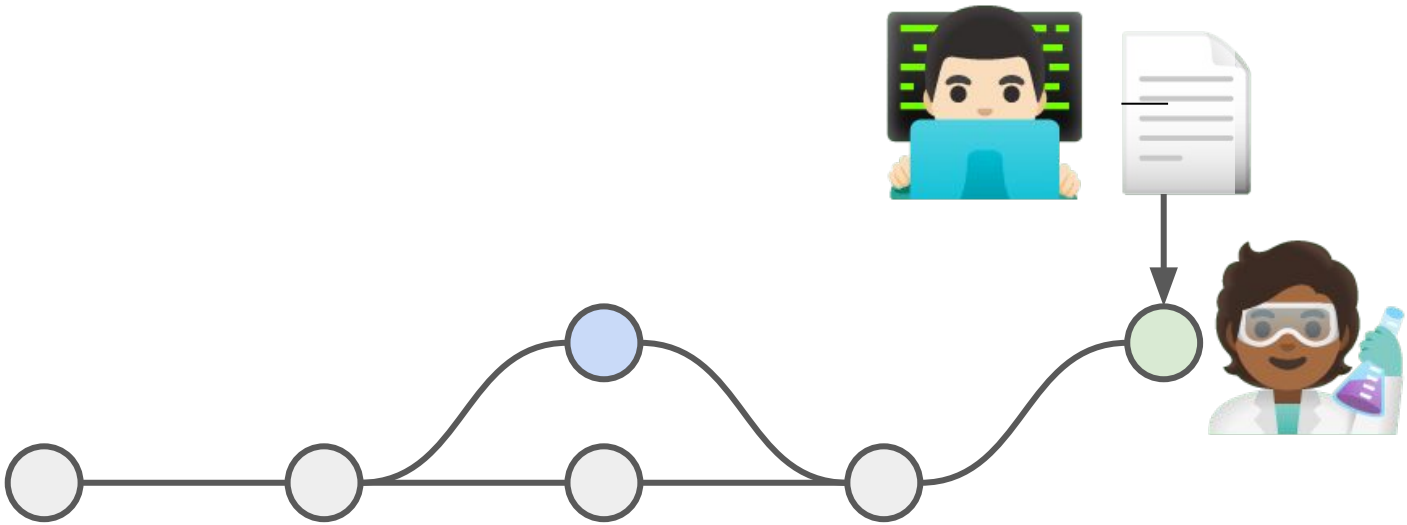


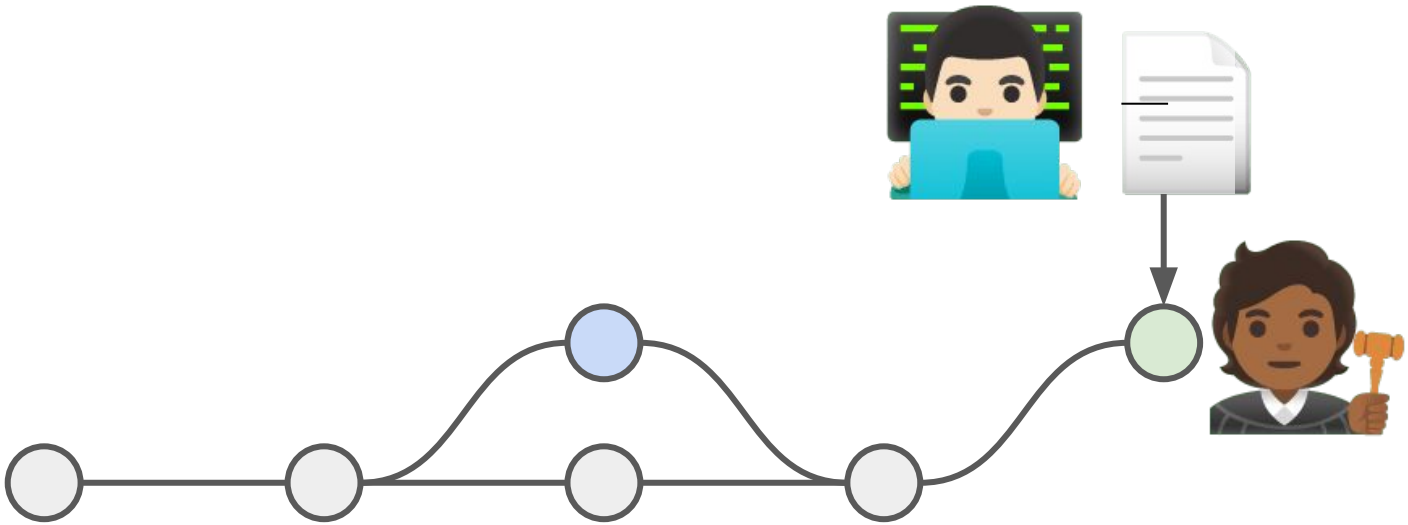


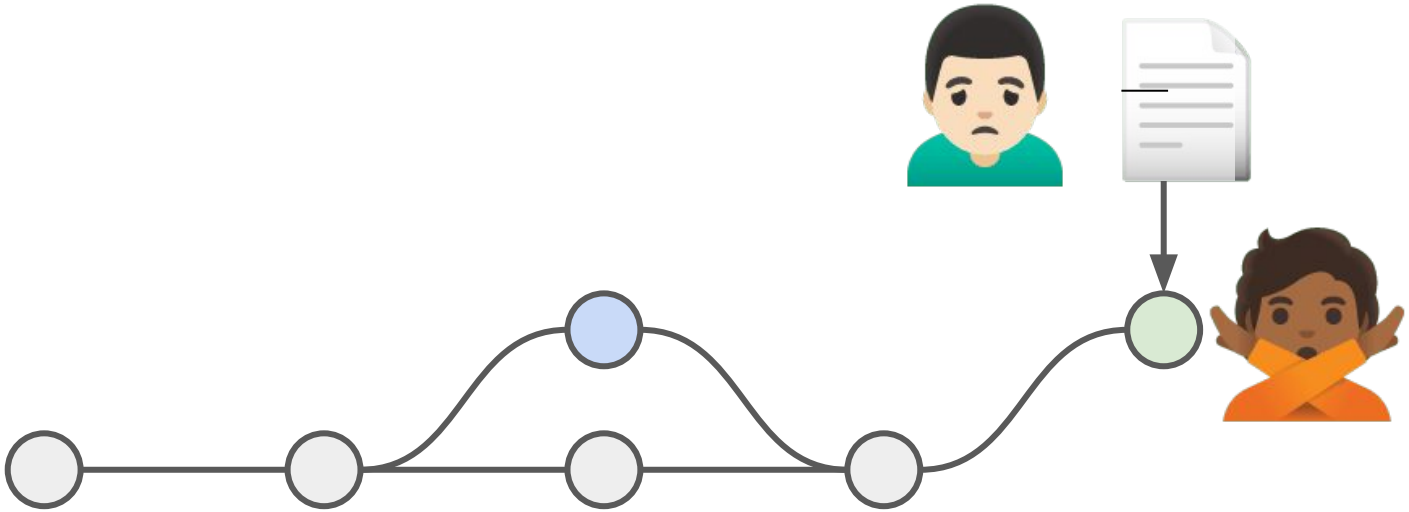


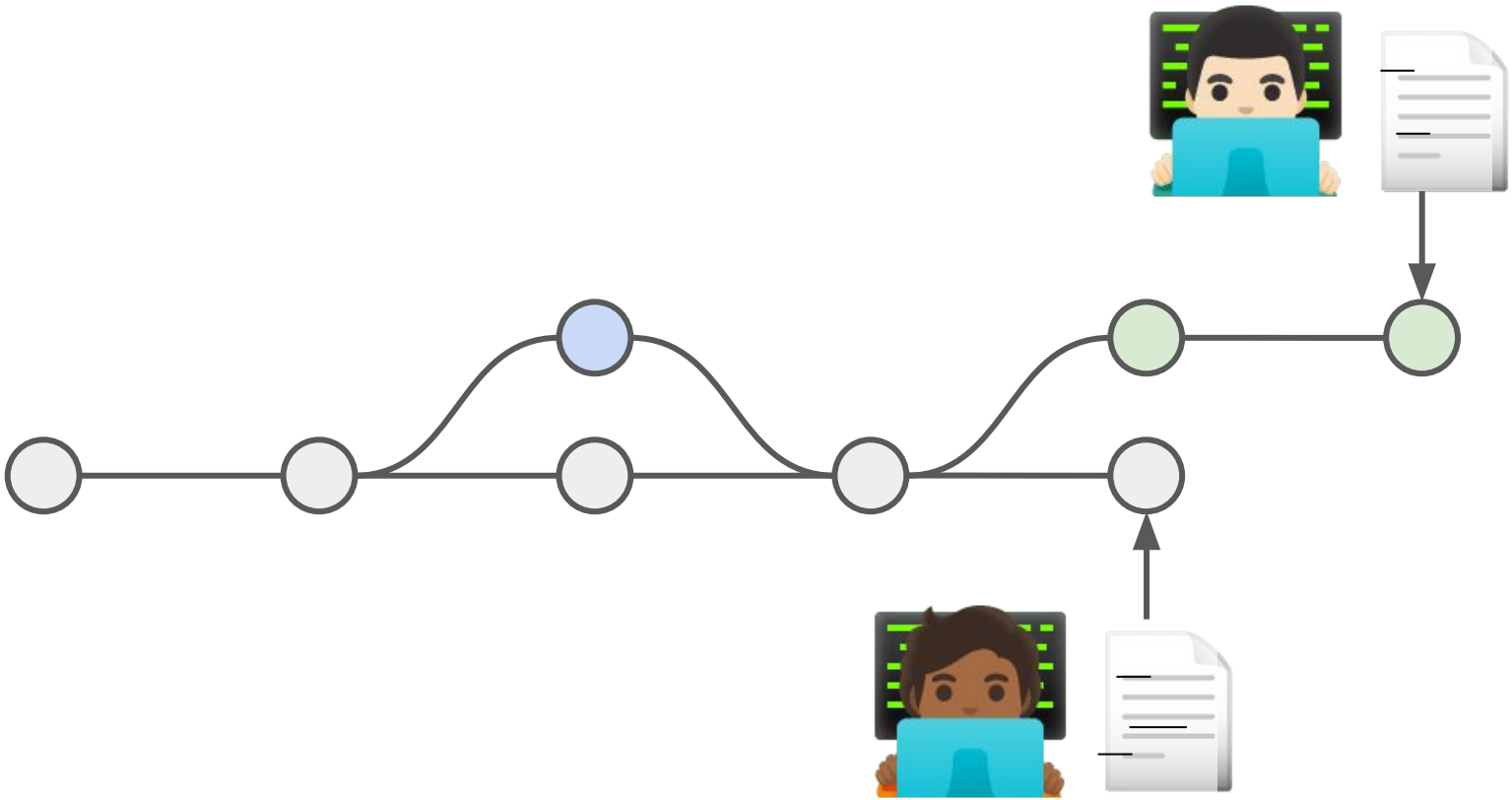

Merge

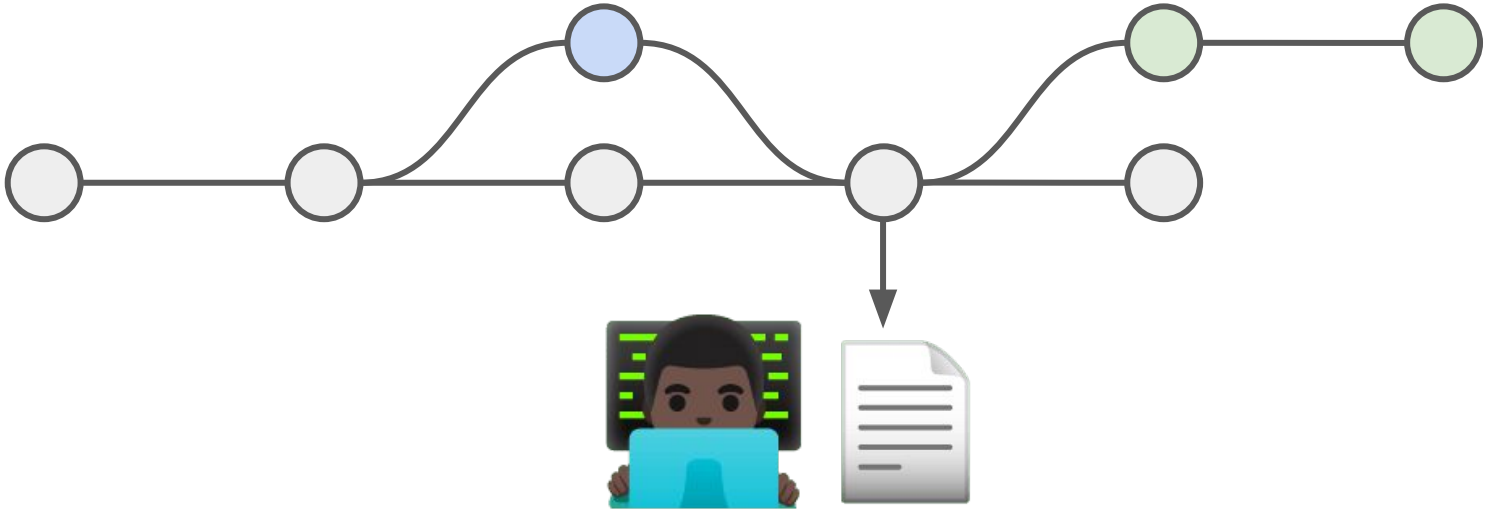








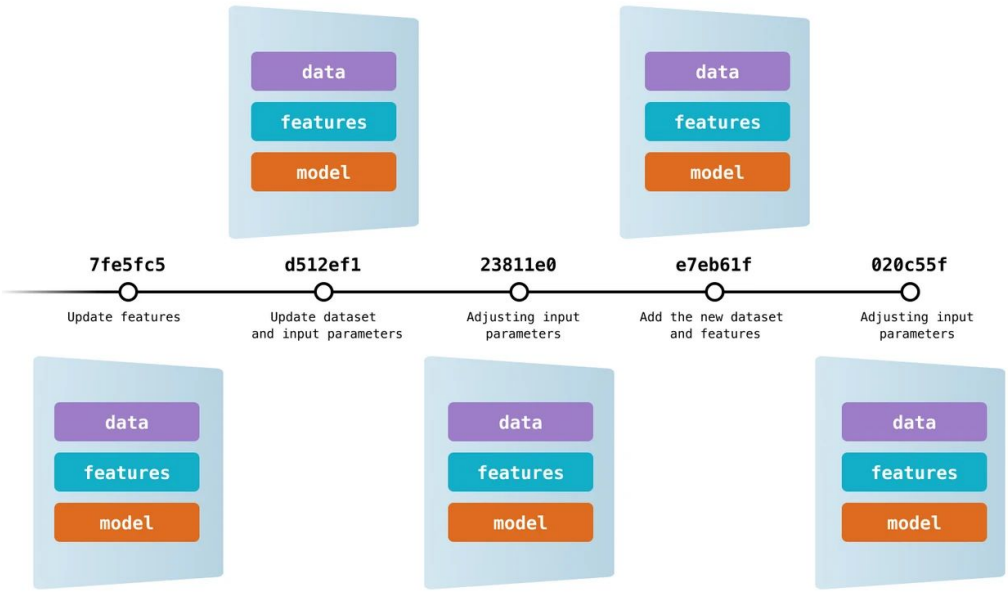


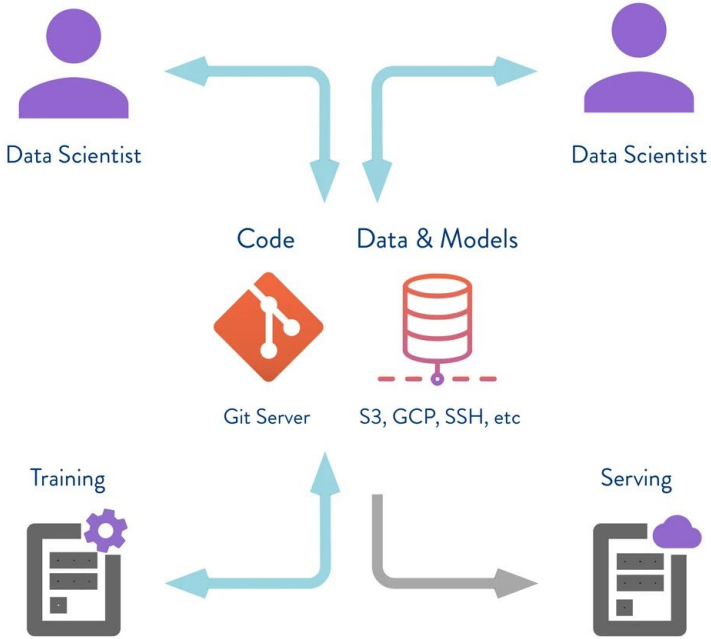


W&B

HUGGING FACE

comet





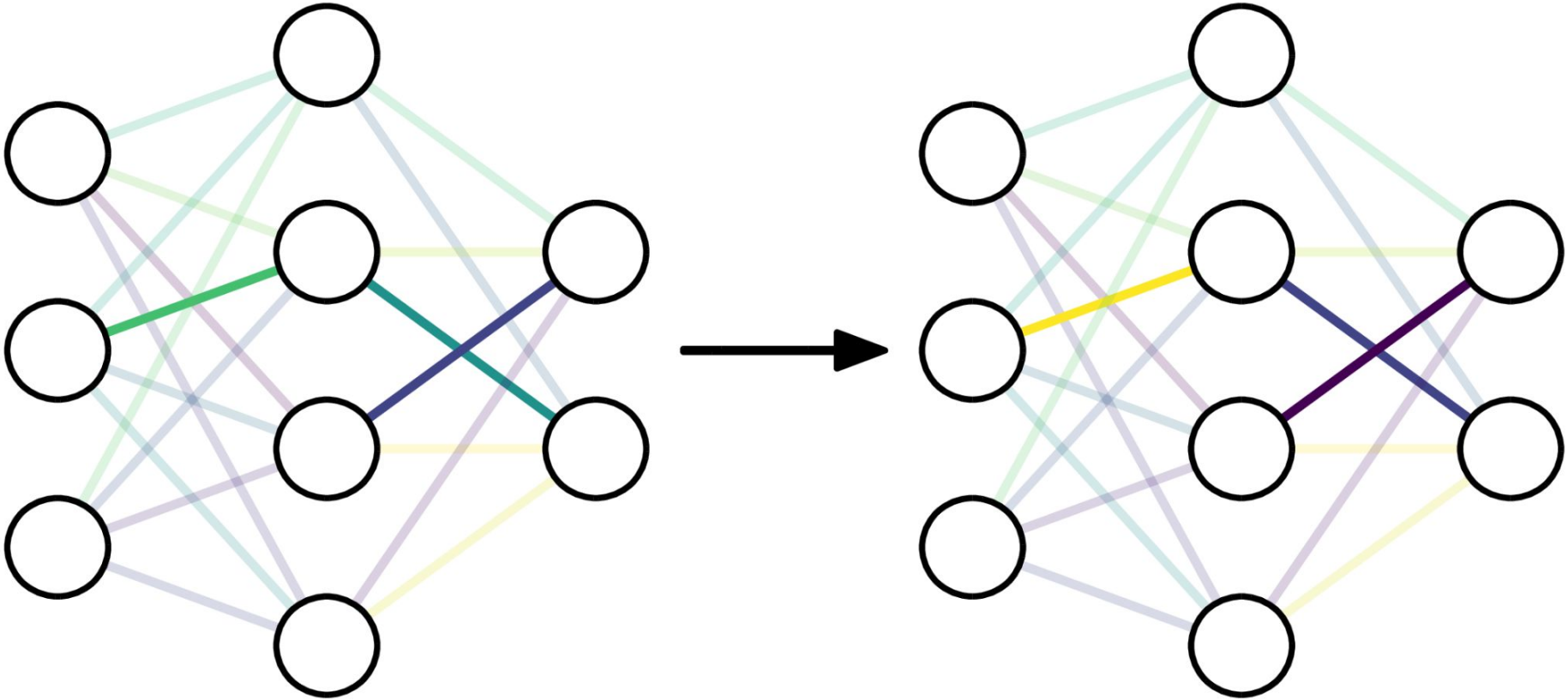
from https://dvc.org/

How can we enable collaborative and continual development of machine learning models?

We need to be able to cheaply communicate **patches** and **merge** updates from different contributors.

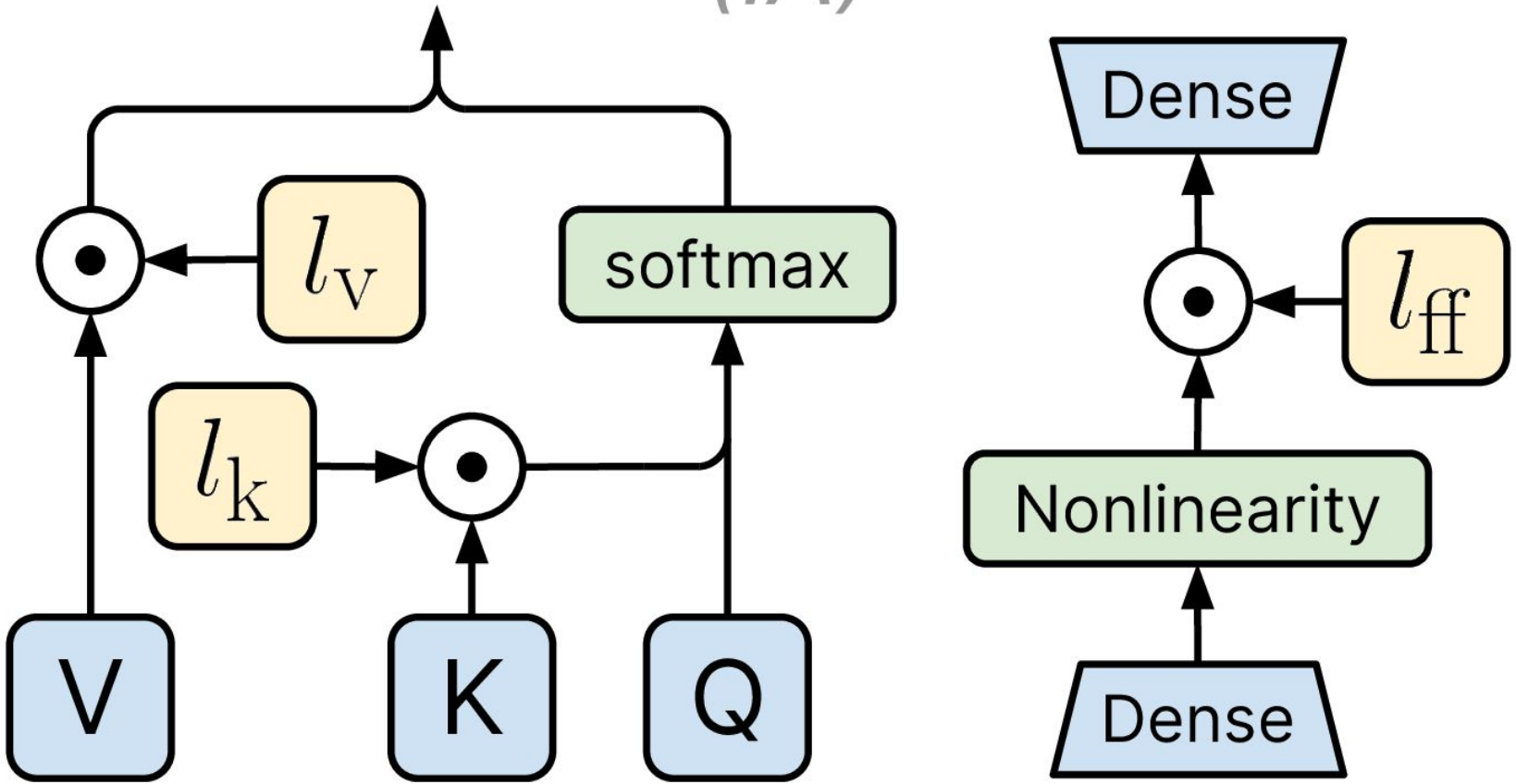
How can we enable collaborative and continual development of machine learning models?

We need to be able to cheaply communicate **patches** and **merge** updates from different contributors.



From *Training Neural Networks with Fixed Sparse Masks* by Sung et al.

$(IA)^3$



From *Few-Shot Parameter-Efficient Fine-Tuning is Better and Cheaper than In-Context Learning* by Liu et al.

$$\mathbf{h}_2^\ell = \text{Dropout}\left(\mathbf{W}_{m_1}^\ell \cdot \mathbf{h}_1^\ell + \mathbf{b}_{m_1}^\ell\right) \quad (1)$$

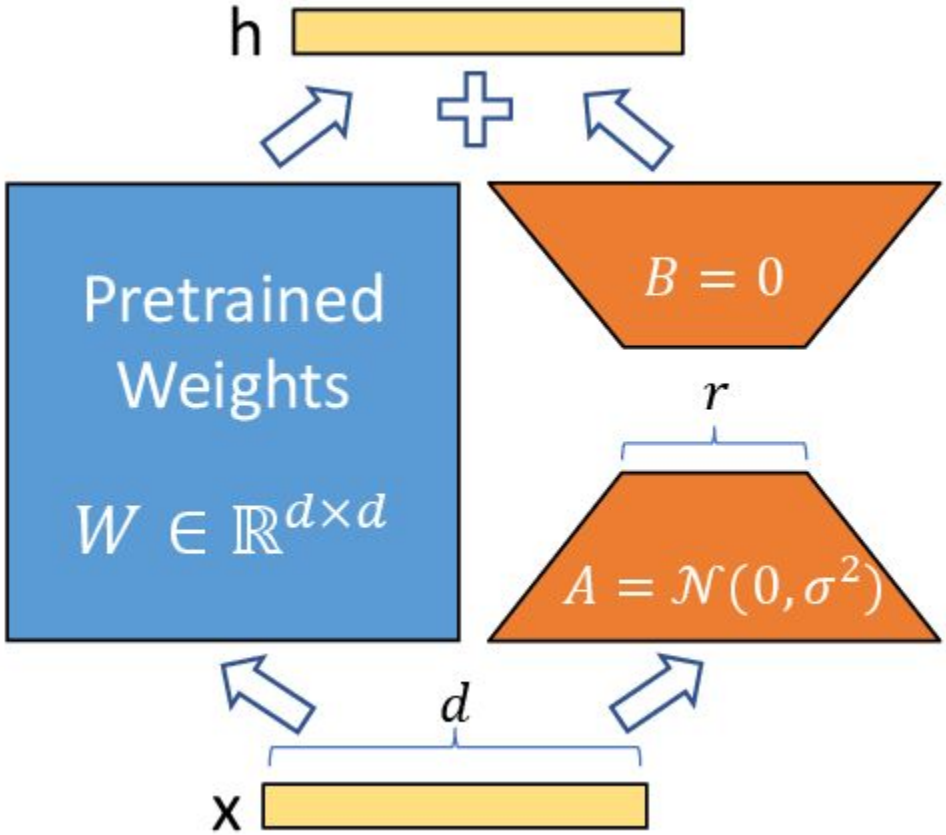
$$\mathbf{h}_3^\ell = \mathbf{g}_{LN_1}^\ell \odot \frac{(\mathbf{h}_2^\ell + \mathbf{x}) - \mu}{\sigma} + \mathbf{b}_{LN_1}^\ell \quad (2)$$

$$\mathbf{h}_4^\ell = \text{GELU}\left(\mathbf{W}_{m_2}^\ell \cdot \mathbf{h}_3^\ell + \mathbf{b}_{m_2}^\ell\right) \quad (3)$$

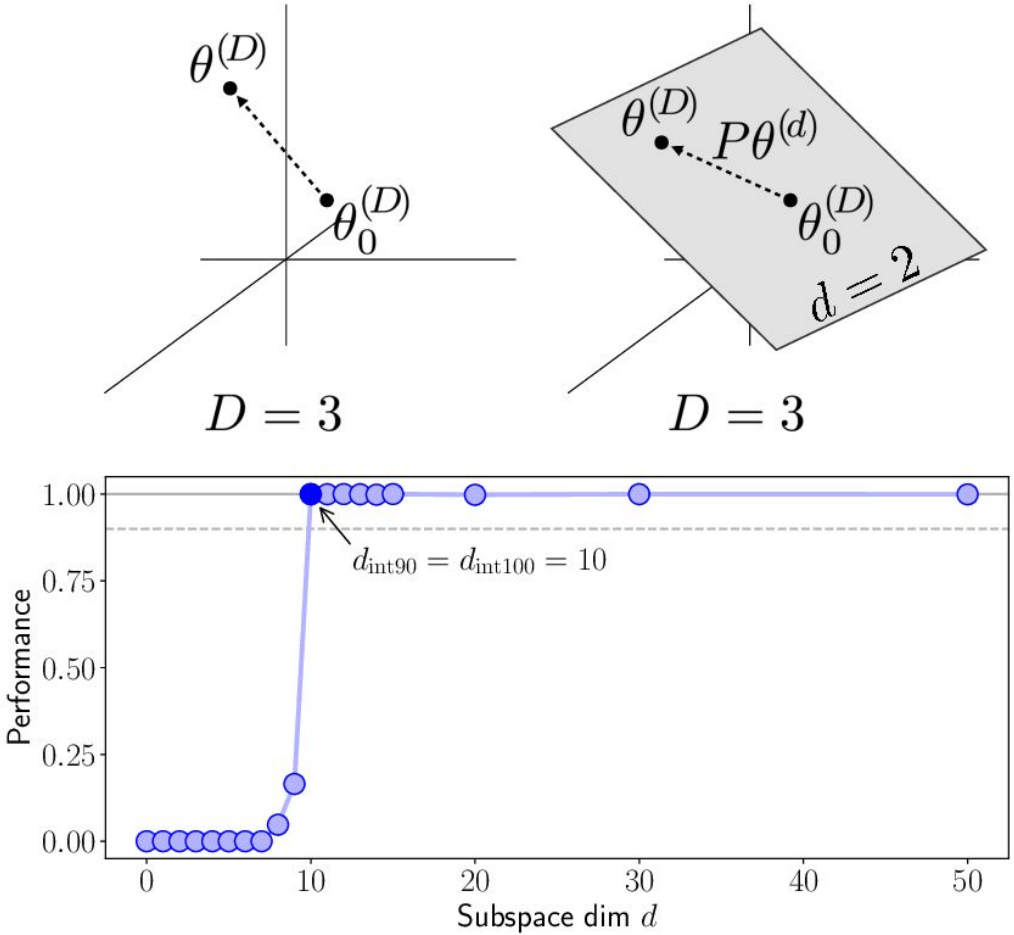
$$\mathbf{h}_5^\ell = \text{Dropout}\left(\mathbf{W}_{m_3}^\ell \cdot \mathbf{h}_4^\ell + \mathbf{b}_{m_3}^\ell\right) \quad (4)$$

$$\text{out}^\ell = \mathbf{g}_{LN_2}^\ell \odot \frac{(\mathbf{h}_5^\ell + \mathbf{h}_3^\ell) - \mu}{\sigma} + \mathbf{b}_{LN_2}^\ell \quad (5)$$

From *BitFit: Simple Parameter-efficient Fine-tuning for Transformer-based Masked Language-models* by Ben Zaken et al.



From *LoRA: Low-Rank Adaptation of Large Language Models* by Hu et al.



From *Measuring the Intrinsic Dimension of Objective Landscapes* by Li et al.

Algorithm 1: Sparse Ternary Compression (STC)

input: flattened tensor $T \in \mathbb{R}^n$, sparsity p

output: sparse ternary tensor $T^* \in \{-\mu, 0, \mu\}^n$

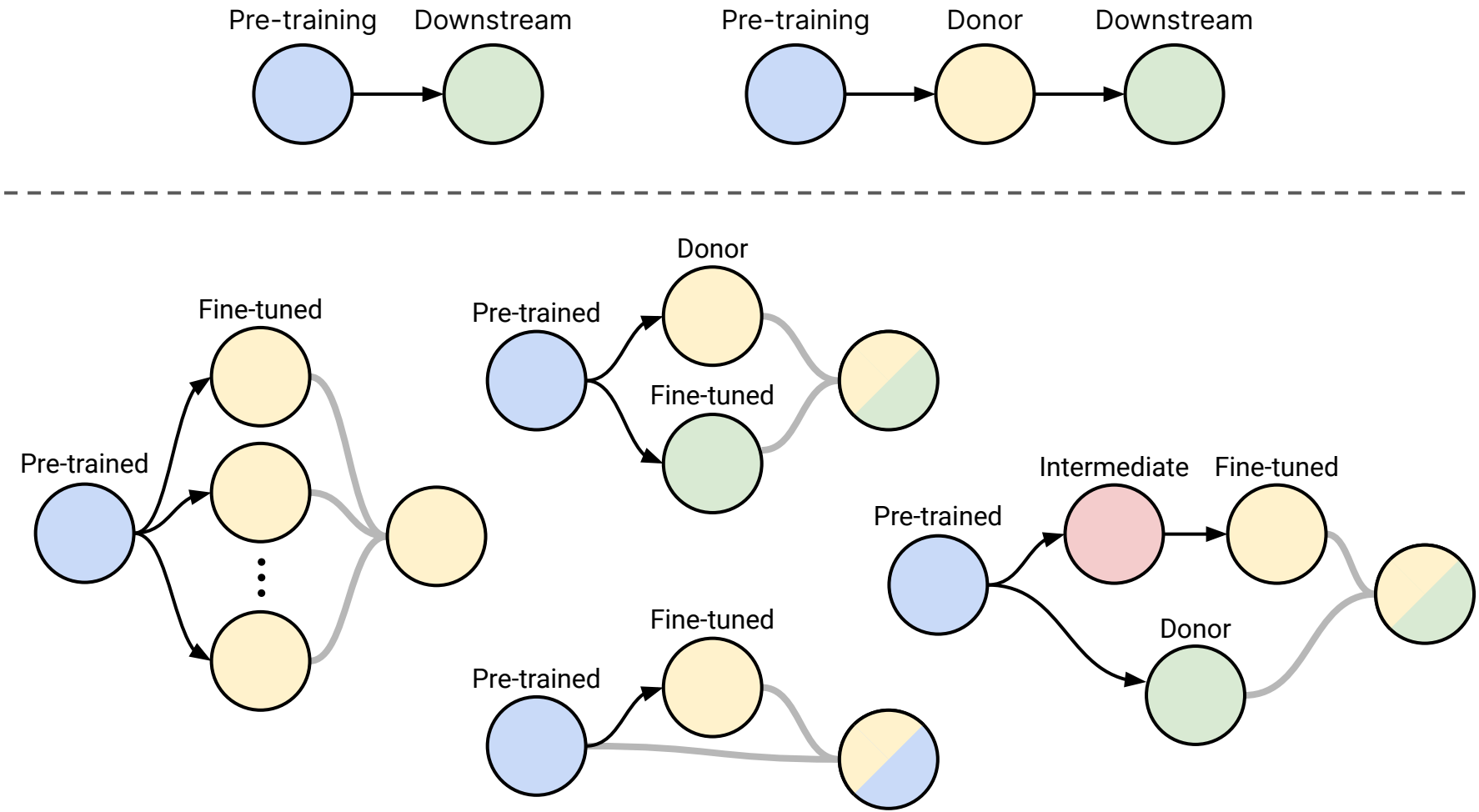
- $k \leftarrow \max(np, 1)$
- $v \leftarrow \text{top}_k(|T|)$
- $\text{mask} \leftarrow (|T| \geq v) \in \{0, 1\}^n$
- $T^{masked} \leftarrow \text{mask} \odot T$
- $\mu \leftarrow \frac{1}{k} \sum_{i=1}^{n} |T_i^{masked}|$

return $T^* \leftarrow \mu \times \text{sign}(T^{masked})$

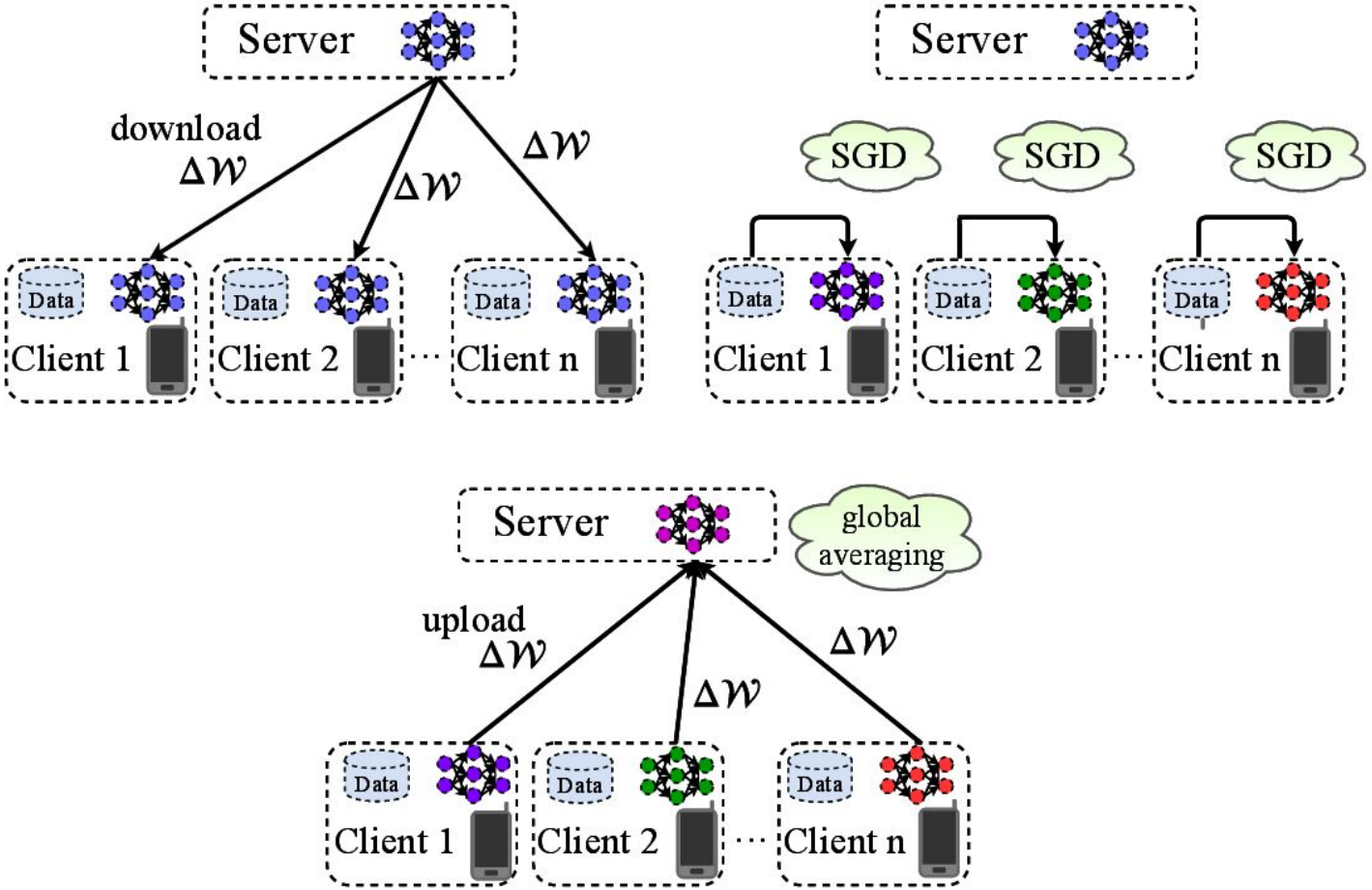
From *Robust and Communication-Efficient Federated Learning from Non-IID Data* by Sattler et al.

How can we enable collaborative and continual development of machine learning models?

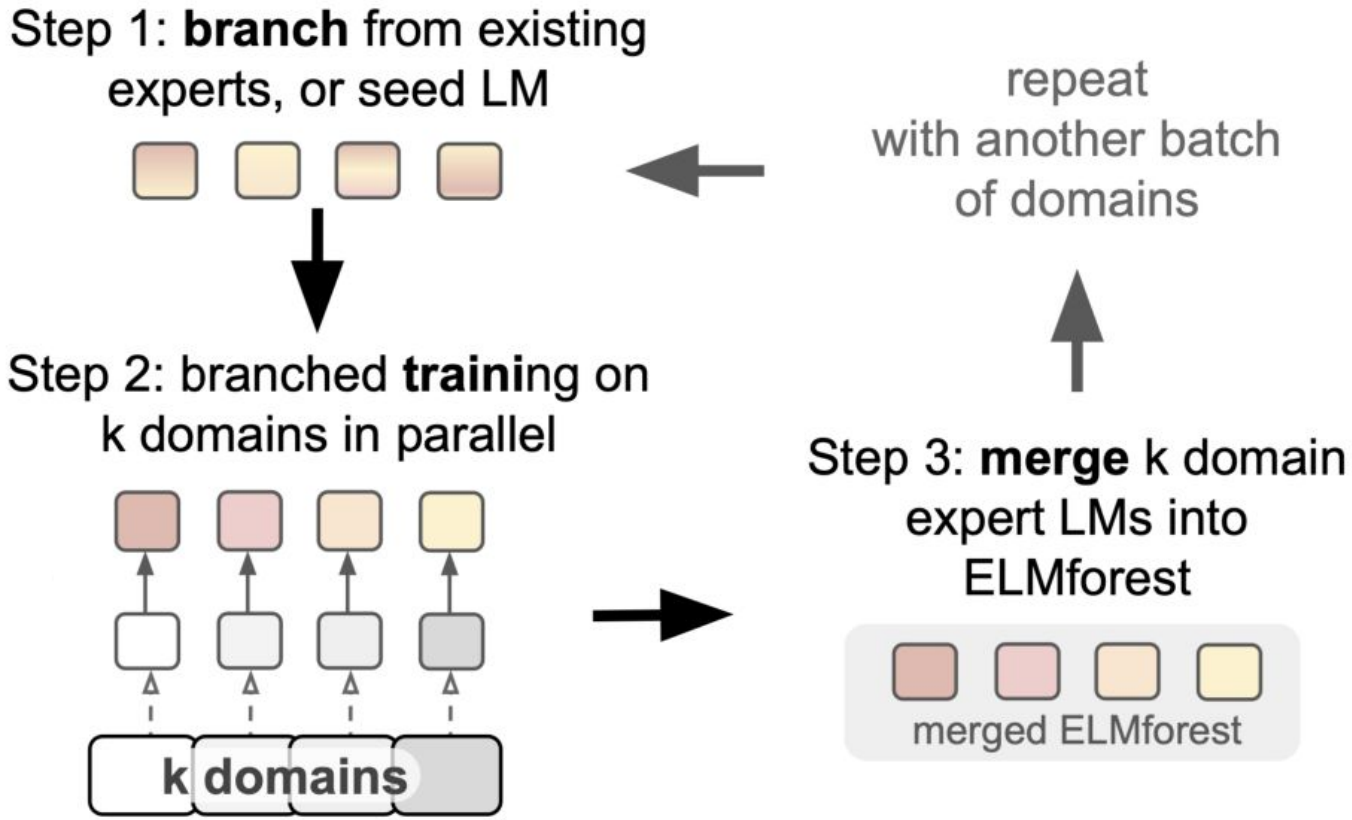
We need to be able to cheaply communicate **patches** and **merge** updates from different contributors.



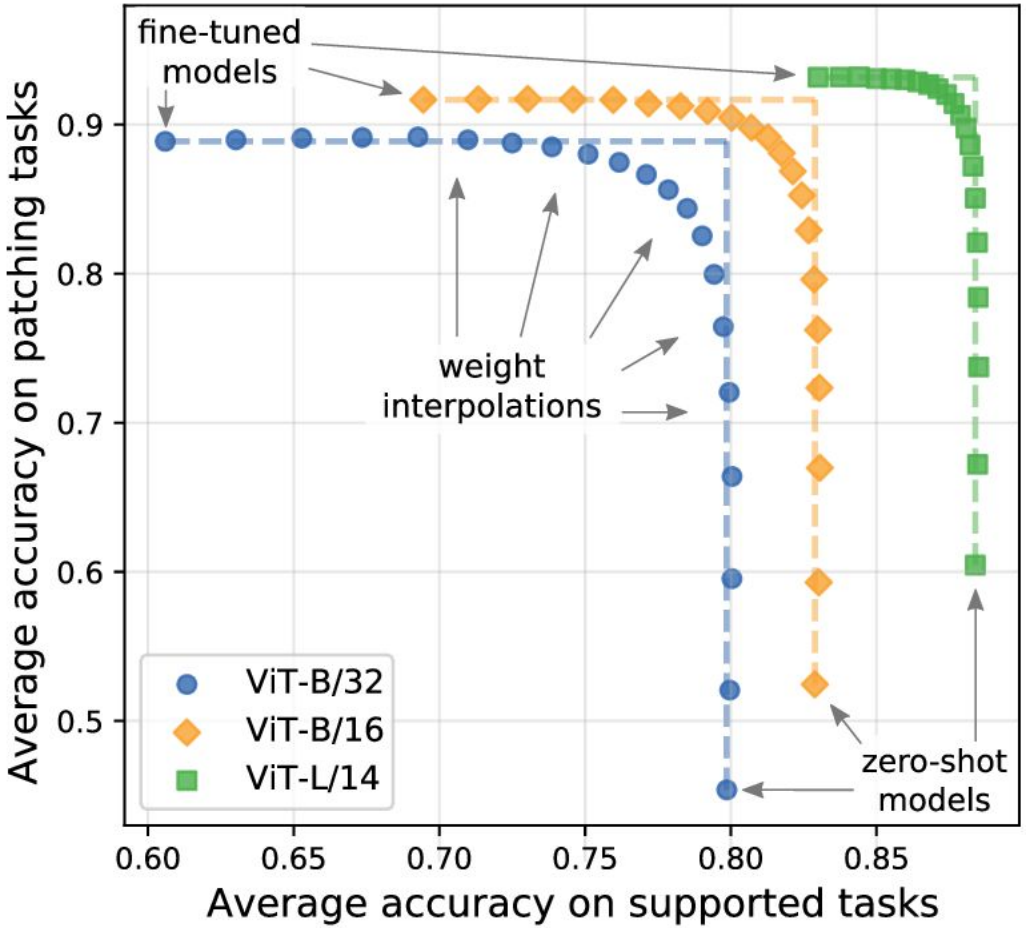
From *Merging Models with Fisher-Weighted Averaging* by Matena and Raffel



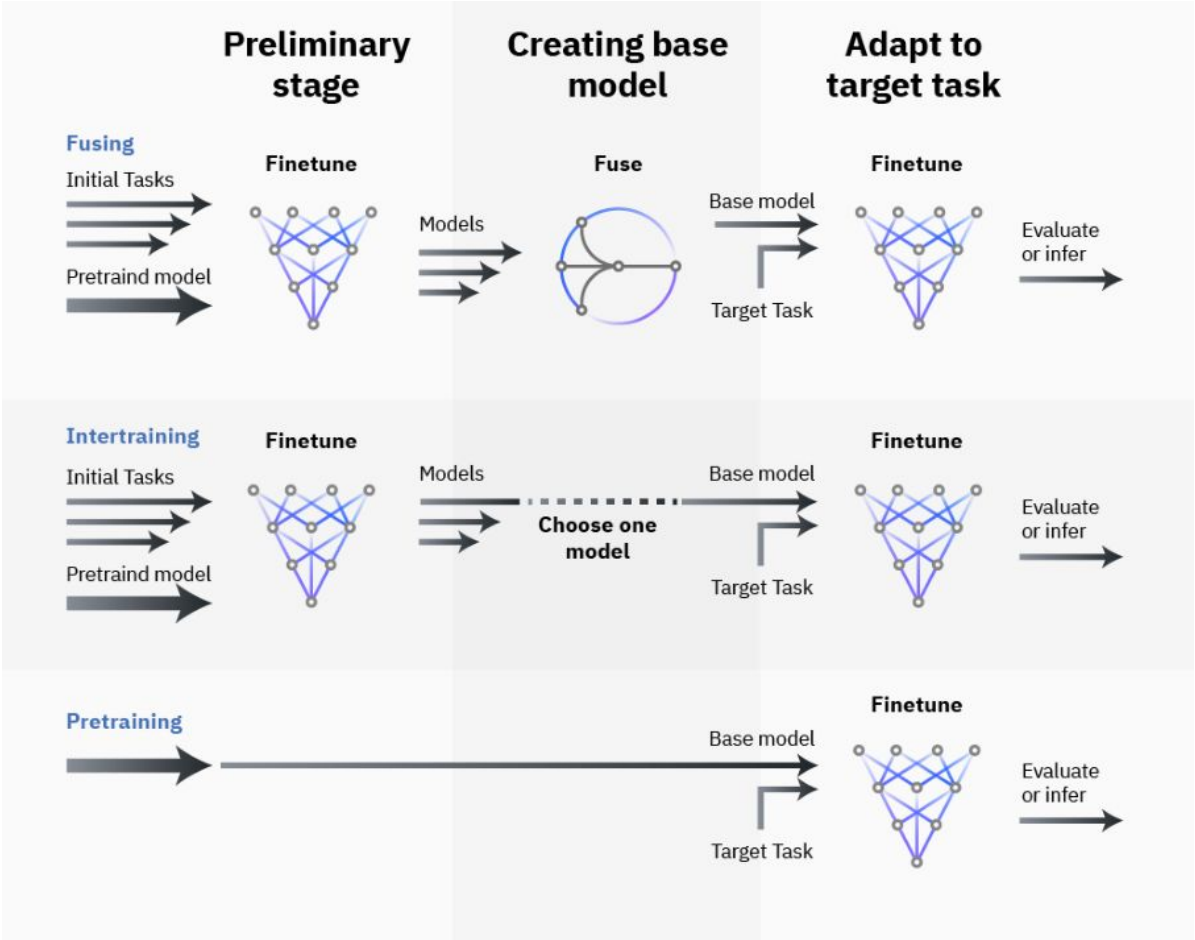
From *Robust and Communication-Efficient Federated Learning from Non-IID Data* by Sattler et al.



From *Branch-Train-Merge: Embarrassingly Parallel Training of Expert Language Models* by Li et al.



From *Patching open-vocabulary models by interpolating weights* by Ilharco et al.

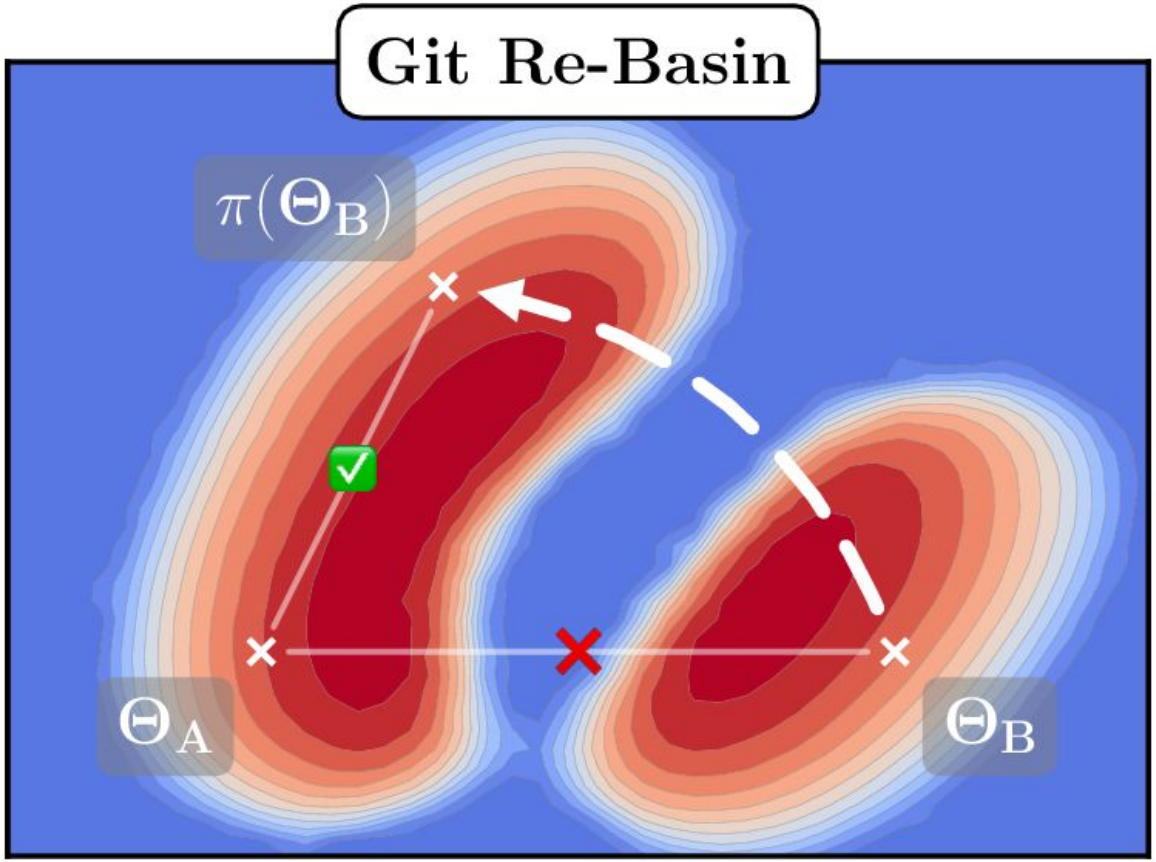


From *Fusing finetuned models for better pretraining* by Choshen et al.

$$\arg\max_{\theta} \sum_{i=1}^{M} \lambda_i \log \mathcal{N}(\theta | \theta_i, \hat{F}_i^{-1})$$

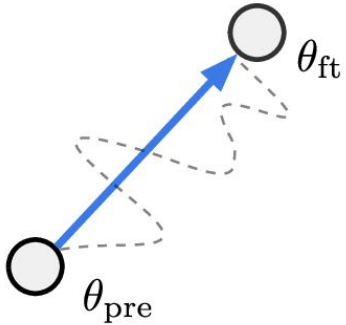
$$\theta^{*(j)} = \frac{\sum_{i=1}^{M} \lambda_i \hat{F}_i^{(j)} \theta_i^{(j)}}{\sum_{i=1}^{M} \lambda_i \hat{F}_i^{(j)}}$$

From *Merging Models with Fisher-Weighted Averaging* by Matena and Raffel



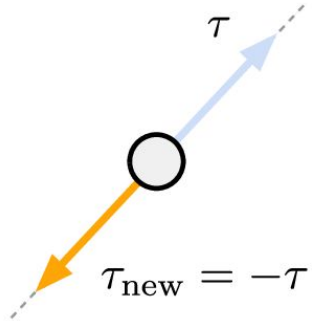
From *Git Re-Basin: Merging Models modulo Permutation Symmetries* by Ainsworth et al.

a) Task vectors



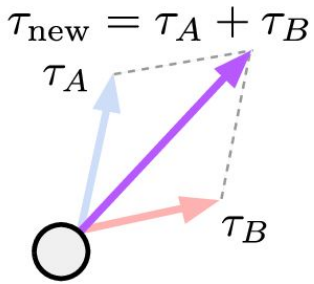
$$\tau = \theta_{\text{ft}} - \theta_{\text{pre}}$$

b) Forgetting via negation



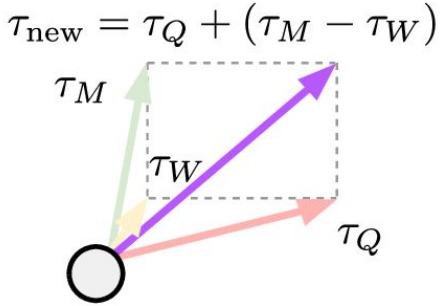
Example: making a language model produce less toxic content

c) Learning via addition



Example: building a multi-task model

d) Task analogies



Example: learning about kings with data from queens, man and woman

From *Editing Models with Task Arithmetic* by Ilharco et al.

How can we enable collaborative and continual development of machine learning models?

We need a system to **track changes and manage contributions** to a model.

What we need(ed)

- **Track changes to a checkpoint.** There are many ways to modify a checkpoint, e.g. updating all parameters, updating a subset of parameters, adding/removing parameters, etc. The system should support all of these operations.

What we need(ed)

- **Track changes to a checkpoint.** There are many ways to modify a checkpoint, e.g. updating all parameters, updating a subset of parameters, adding/removing parameters, etc. The system should support all of these operations.
- **An on-disk format for changes.** This format should only store the information required to reconstruct the change and should support all update types.

What we need(ed)

- **Track changes to a checkpoint.** There are many ways to modify a checkpoint, e.g. updating all parameters, updating a subset of parameters, adding/removing parameters, etc. The system should support all of these operations.
- **An on-disk format for changes.** This format should only store the information required to reconstruct the change and should support all update types.
- **Restoring the previous state of a checkpoint.** The system should be able to rapidly restore the state of the model at a certain point of history.

What we need(ed)

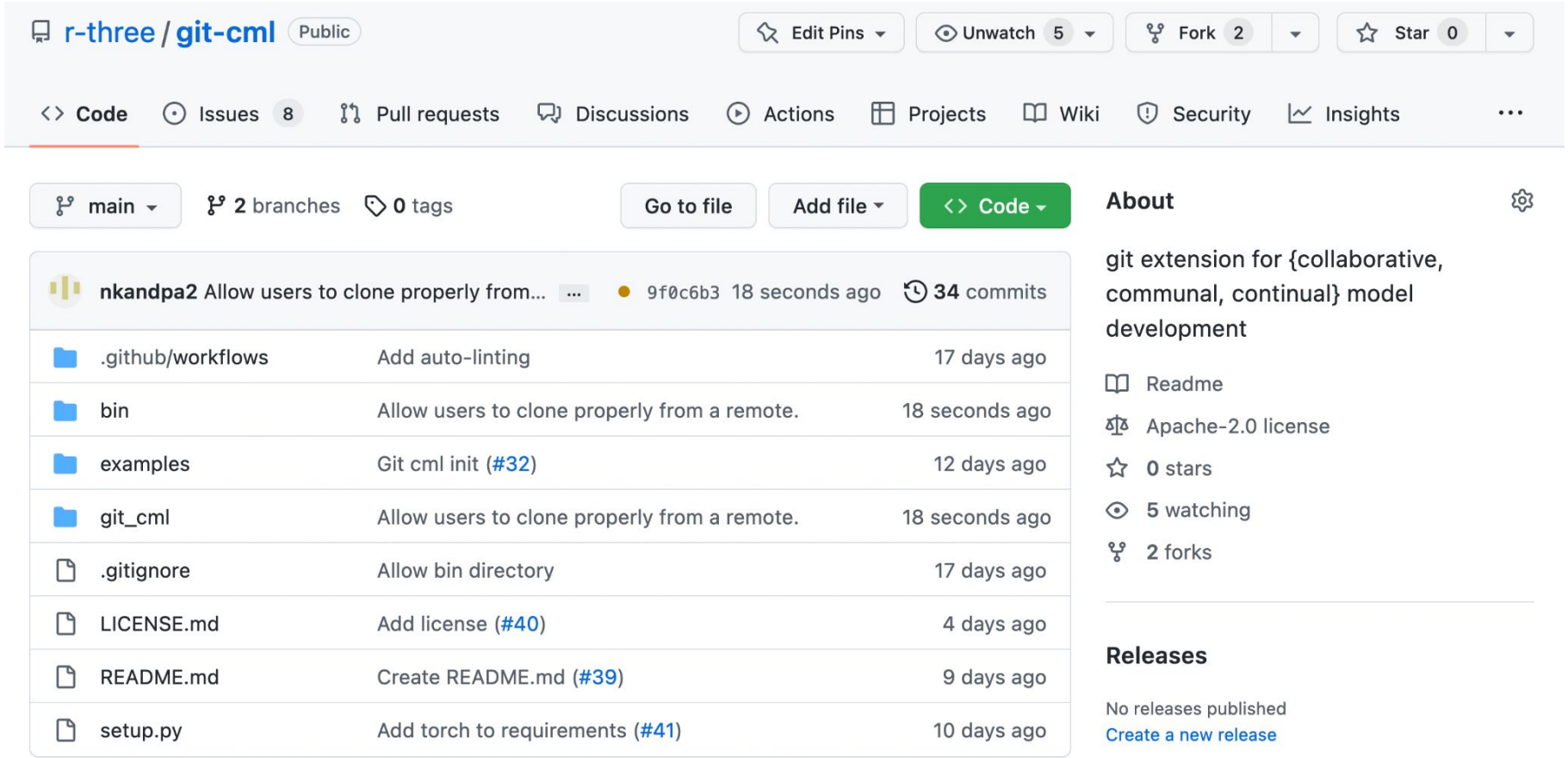
- **Track changes to a checkpoint.** There are many ways to modify a checkpoint, e.g. updating all parameters, updating a subset of parameters, adding/removing parameters, etc. The system should support all of these operations.
- **An on-disk format for changes.** This format should only store the information required to reconstruct the change and should support all update types.
- **Restoring the previous state of a checkpoint.** The system should be able to rapidly restore the state of the model at a certain point of history.
- **Communicate/receive history.** There should be a way for me to communicate my local history to a model to someone/someplace else, as well as bring in the changes to the model someone else has proposed.

What we need(ed)

- **Track changes to a checkpoint.** There are many ways to modify a checkpoint, e.g. updating all parameters, updating a subset of parameters, adding/removing parameters, etc. The system should support all of these operations.
- **An on-disk format for changes.** This format should only store the information required to reconstruct the change and should support all update types.
- **Restoring the previous state of a checkpoint.** The system should be able to rapidly restore the state of the model at a certain point of history.
- **Communicate/receive history.** There should be a way for me to communicate my local history to a model to someone/someplace else, as well as bring in the changes to the model someone else has proposed.
- **Identifying and possibly resolving merge conflicts.** When there are conflicting changes created in parallel, the system should fail or (ideally, eventually) try to resolve them.

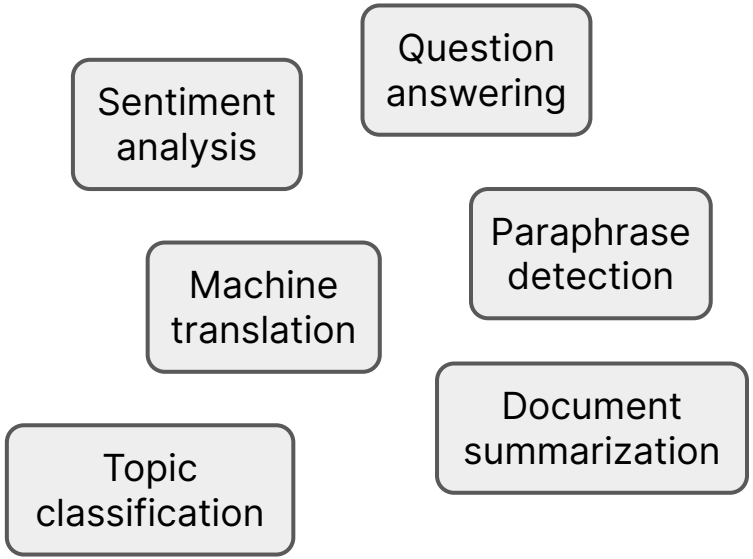
What we need(ed)

- **Track changes to a checkpoint.** There are many ways to modify a checkpoint, e.g. updating all parameters, updating a subset of parameters, adding/removing parameters, etc. The system should support all of these operations.
- **An on-disk format for changes.** This format should only store the information required to reconstruct the change and should support all update types.
- **Restoring the previous state of a checkpoint.** The system should be able to rapidly restore the state of the model at a certain point of history.
- **Communicate/receive history.** There should be a way for me to communicate my local history to a model to someone/someplace else, as well as bring in the changes to the model someone else has proposed.
- **Identifying and possibly resolving merge conflicts.** When there are conflicting changes created in parallel, the system should fail or (ideally, eventually) try to resolve them.
- **Easy to go from training → version control system.** We don't expect contributors to directly create patches; we expect them to use version control-aware training code that needs to interact with the system to create patches, etc.


r-three / git-cml
Public
Edit Pins
Unwatch 5
Fork 2
Star 0
Code
Issues 8
Pull requests
Discussions
Actions
Projects
Wiki
Security
Insights
main
2 branches
0 tags
Go to file
Add file
Code
nkandpa2 Allow users to clone properly from...
9f0c6b3 18 seconds ago
34 commits
.github/workflows
Add auto-linting
17 days ago
bin
Allow users to clone properly from a remote.
18 seconds ago
examples
Git cml init (#32)
12 days ago
git_cml
Allow users to clone properly from a remote.
18 seconds ago
.gitignore
Allow bin directory
17 days ago
LICENSE.md
Add license (#40)
4 days ago
README.md
Create README.md (#39)
9 days ago
setup.py
Add torch to requirements (#41)
10 days ago
About
git extension for {collaborative, communal, continual} model development
Readme
Apache-2.0 license
0 stars
5 watching
2 forks
Releases
No releases published
Create a new release

How can we enable collaborative and continual development of machine learning models?

Let’s try it by focusing on a specific architecture.


Question answering
Sentiment analysis
Paraphrase detection
Machine translation
Document summarization
Topic classification

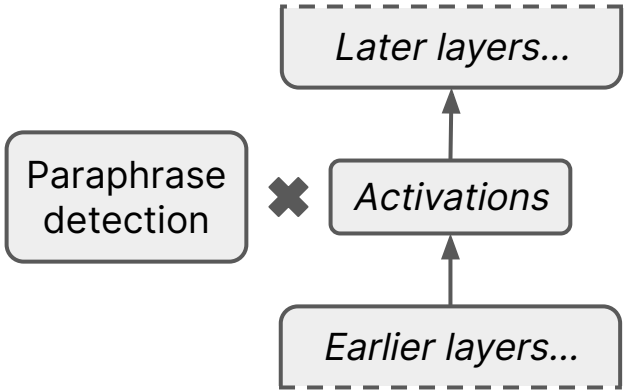
Sentiment analysis

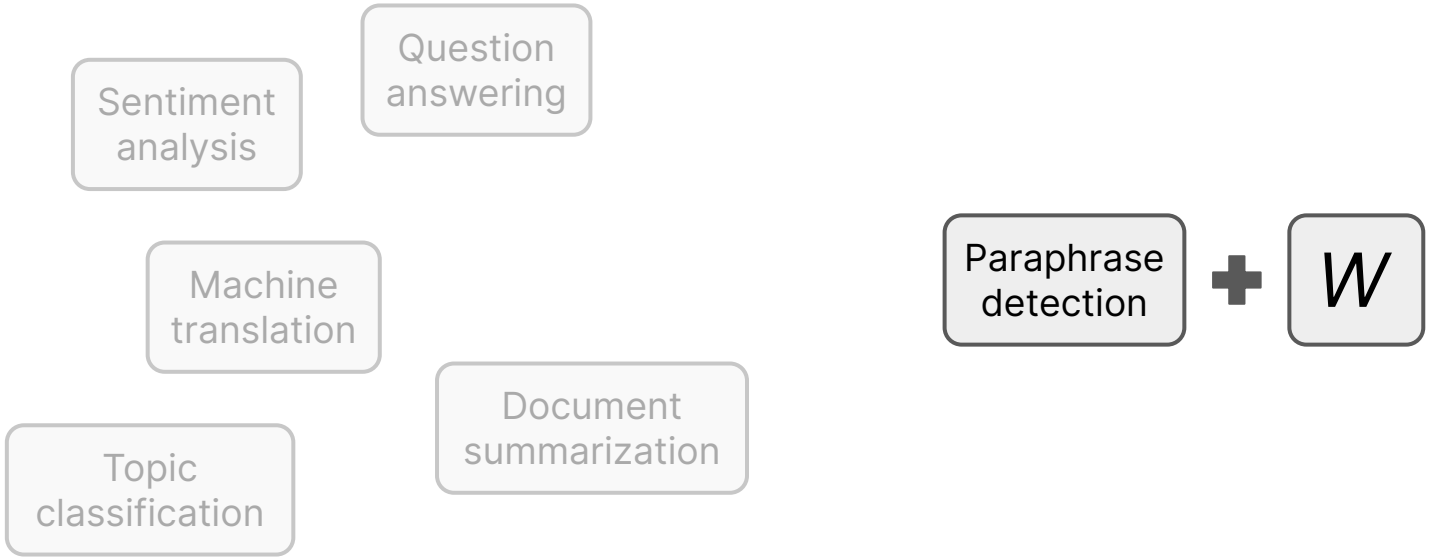
Question answering

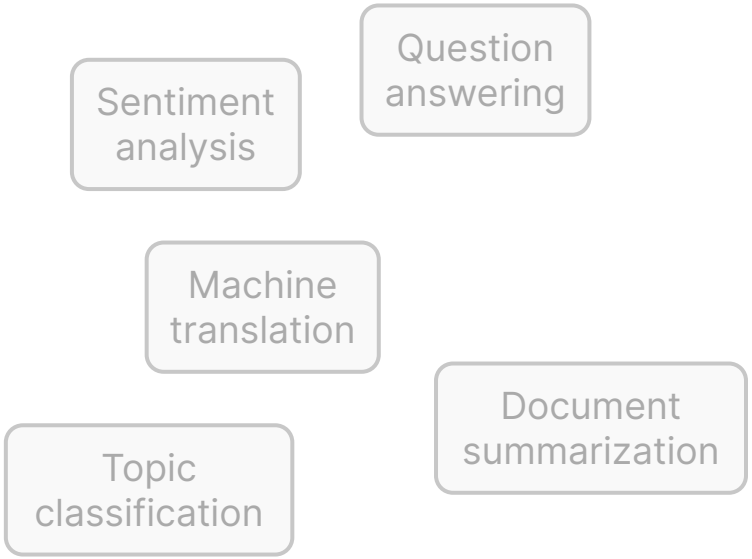
Machine translation

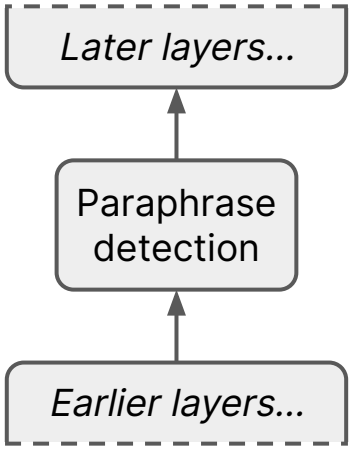
Document summarization

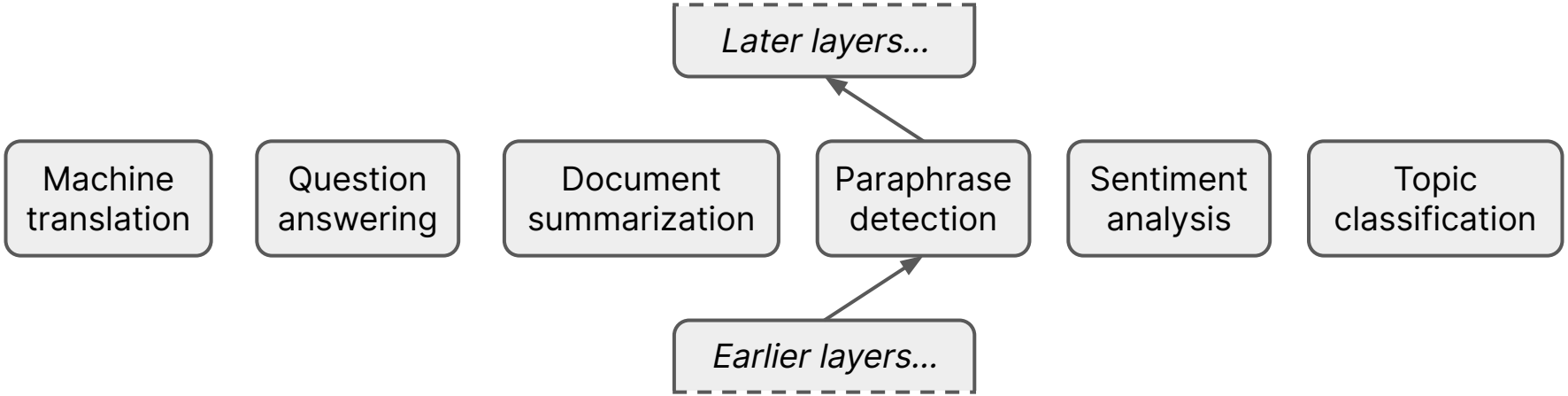
Topic classification

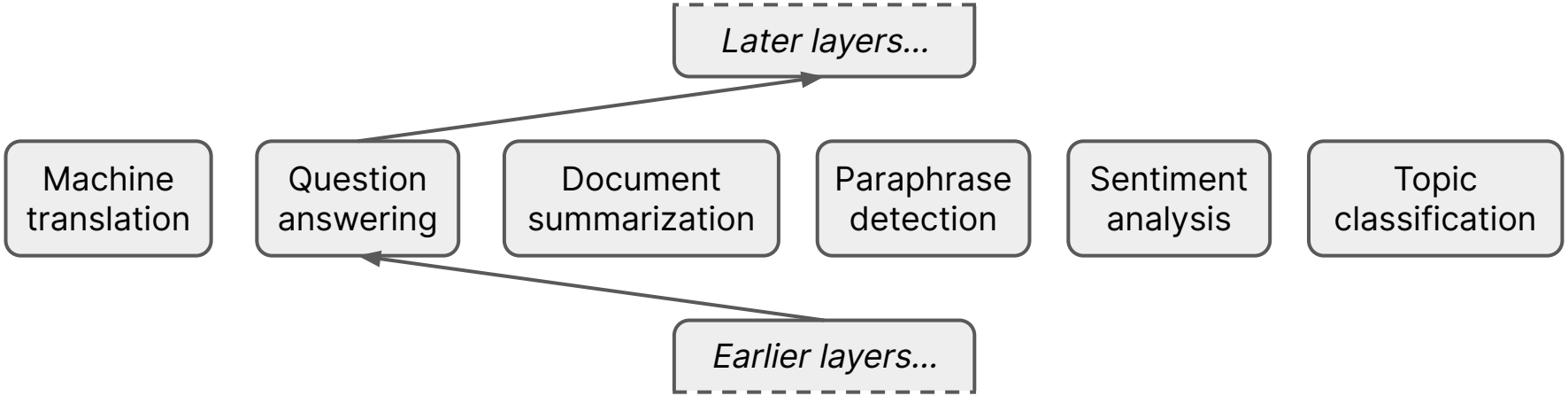


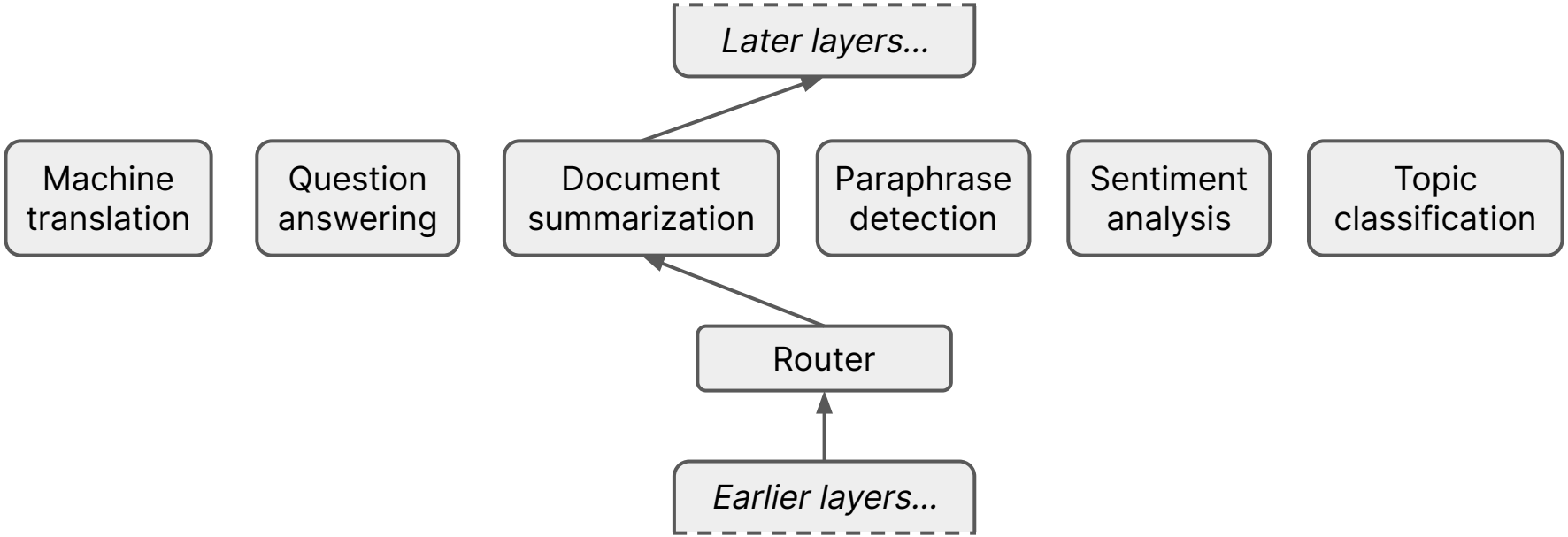

Question answering
Sentiment analysis
Machine translation
Document summarization
Topic classification
Paraphrase detection
W

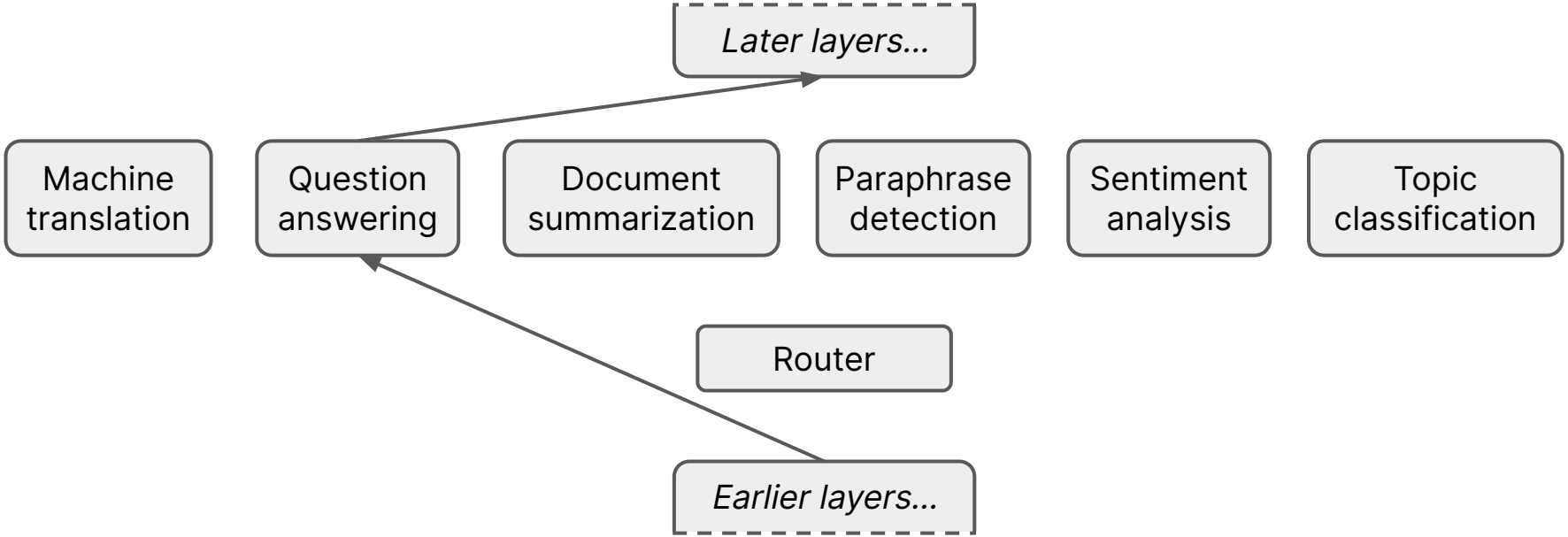

Question answering
Sentiment analysis
Machine translation
Document summarization
Topic classification


Later layers...
Paraphrase detection
Earlier layers...


Later layers...
Machine translation
Question answering
Document summarization
Paraphrase detection
Sentiment analysis
Topic classification
Earlier layers...


Later layers...
Machine translation
Question answering
Document summarization
Paraphrase detection
Sentiment analysis
Topic classification
Earlier layers...


Later layers...
Machine translation
Question answering
Document summarization
Paraphrase detection
Sentiment analysis
Topic classification
Router
Earlier layers...


Later layers...
Machine translation
Question answering
Document summarization
Paraphrase detection
Sentiment analysis
Topic classification
Router
Earlier layers...

Why I think this would be a good model

- **It would probably work.** Parameter-efficient fine-tuning methods work well. We can probably get the learned routing to work reasonably well.

Why I think this would be a good model

- **It would probably work.** Parameter-efficient fine-tuning methods work well. We can probably get the learned routing to work reasonably well.
- **It would be useful.** The model would be able to perform tons of tasks out-of-the-box. This would make it convenient and compelling to use.

Why I think this would be a good model

- **It would probably work.** Parameter-efficient fine-tuning methods work well. We can probably get the learned routing to work reasonably well.
- **It would be useful.** The model would be able to perform tons of tasks out-of-the-box. This would make it convenient and compelling to use.
- **Most version control operations are trivial.** People can add or update specific adapters, which results in cheap-to-communicate updates that are trivial to merge. Testing would be easy too - just evaluate performance on the task for the updated adapter. Updating/adding adapters is probably how most people would contribute.

Why I think this would be a good model

- **It would probably work.** Parameter-efficient fine-tuning methods work well. We can probably get the learned routing to work reasonably well.
- **It would be useful.** The model would be able to perform tons of tasks out-of-the-box. This would make it convenient and compelling to use.
- **Most version control operations are trivial.** People can add or update specific adapters, which results in cheap-to-communicate updates that are trivial to merge. Testing would be easy too - just evaluate performance on the task for the updated adapter. Updating/adding adapters is probably how most people would contribute.
- **Could occasionally update the router or backbone.** These can be seen as minor/major version updates...

Why I think this would be a good model

- **It would probably work.** Parameter-efficient fine-tuning methods work well. We can probably get the learned routing to work reasonably well.
- **It would be useful.** The model would be able to perform tons of tasks out-of-the-box. This would make it convenient and compelling to use.
- **Most version control operations are trivial.** People can add or update specific adapters, which results in cheap-to-communicate updates that are trivial to merge. Testing would be easy too - just evaluate performance on the task for the updated adapter. Updating/adding adapters is probably how most people would contribute.
- **Could occasionally update the router or backbone.** These can be seen as minor/major version updates...
- **Could consider adapters beyond task-level.** For example, we could say each task has some particular domain and language, and have separate domain and language adapter sets.

Why I think this would be a good model

- **It would probably work.** Parameter-efficient fine-tuning methods work well. We can probably get the learned routing to work reasonably well.
- **It would be useful.** The model would be able to perform tons of tasks out-of-the-box. This would make it convenient and compelling to use.
- **Most version control operations are trivial.** People can add or update specific adapters, which results in cheap-to-communicate updates that are trivial to merge. Testing would be easy too - just evaluate performance on the task for the updated adapter. Updating/adding adapters is probably how most people would contribute.
- **Could occasionally update the router or backbone.** These can be seen as minor/major version updates...
- **Could consider adapters beyond task-level.** For example, we could say each task has some particular domain and language, and have separate domain and language adapter sets.
- **Some precedent in AdapterHub.** This model could be seen as a next-generation version; here, adapters ship with the model, routing can be learned, and there is a more principled way of tracking changes.

Join us!

https://bit.ly/cccml-community

Give me feedback:

https://bit.ly/colin-talk-feedback