Academic Paper Writing in Artificial Intelligence

Method Section (II): Loss Functions and Training Details

Instructor: Ying-Jia Lin
Octobor 15th, 2026

Couse Website
Download slide here!

Outline

- Week 5 Assignment Summary (10 min)
- What the Method section is for (10 min)
- Problem Formulation in **3 steps** (30 min)
- **Your Turn for revision and new assignment (35 min, including break)**
- Model Architecture in **2 steps** (25 min)
- **Your Turn (40 min)**

Week 5 Assignment Summary (1/2)

- Some information was missing.
  - Our overall framework is shown in Figure 1. The input is first processed by the module in Section 3.1, followed by **hidden state aggregations**. Then Qwen 3.8 (Q Team, 2026) takes the inputs and generate the reports.

Week 5 Assignment Summary (1/2)

- Some information was missing.
  - Our overall framework is shown in Figure 1. The input is first processed by the module in Section 3.1, followed by **hidden state aggregations**. Then Qwen 3.8 (Q Team, 2026) takes the inputs and generate the reports.

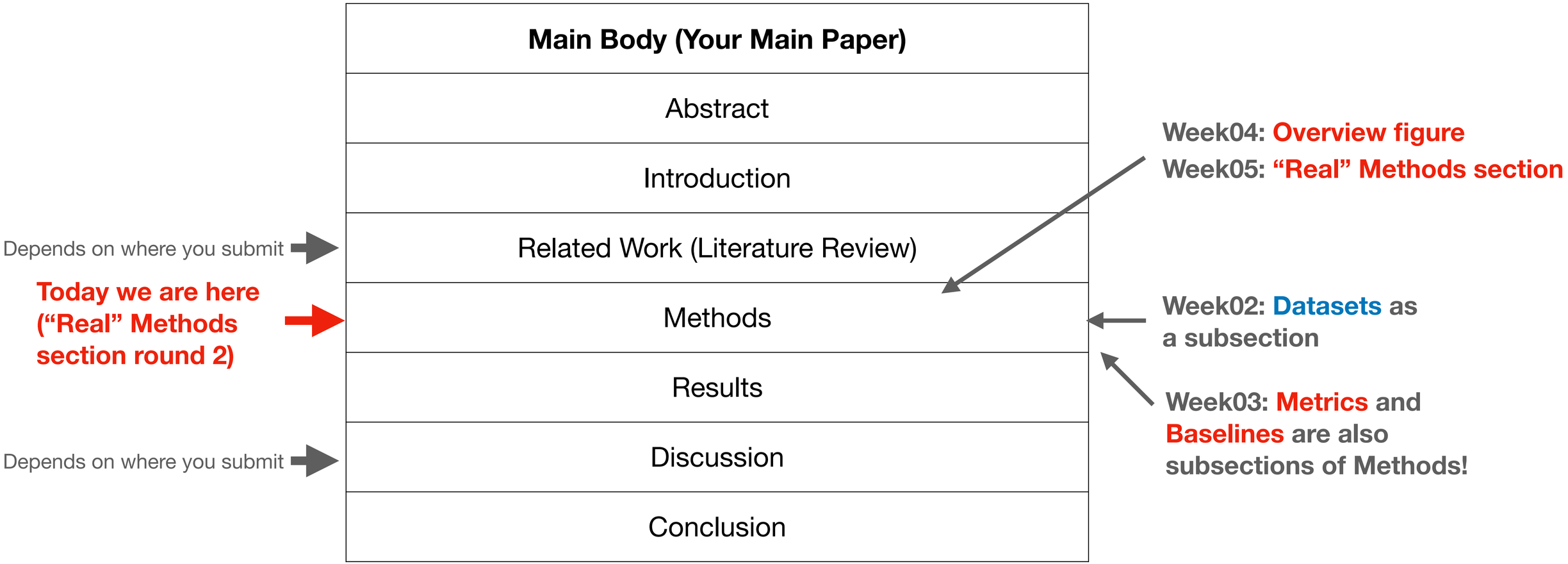


The readers do not know this!

Week 5 Assignment Summary (2/2)

- Some information was not consistent with the overview figure.
  - Figure 1: hidden state aggregations
  - Methods section says: **hidden state fusion**

Where we are


Main Body (Your Main Paper)
Abstract
Introduction
Related Work (Literature Review)
Methods
Results
Discussion
Conclusion
Depends on where you submit
Today we are here ("Real" Methods section round 2)
Depends on where you submit
Week04: Overview figure
Week05: "Real" Methods section
Week02: Datasets as a subsection
Week03: Metrics and Baselines are also subsections of Methods!

Two Examples that Datasets not in Methods

- Li, Mingjie, et al. "Advancing In-Context Learning for Efficient and Stable Medical Report Generation." IEEE Transactions on Pattern Analysis and Machine Intelligence (2026).
- Yan, Sixing, et al. "Learning self-critiquing mechanisms for region-guided chest x-ray report generation." International Conference on Learning Representations. Vol. 2026. 2026.

Loss Functions

Three steps to write this subsection

1. Announce — state the objective, and say how many parts it has.

2. Explain — one clause per term: what it is for.

3. Justify — every number and every choice you made.

Three steps to write this subsection

1. Announce — state the objective, and say how many parts it has.

2. Explain — one clause per term: what it is for.

3. Justify — every number and every choice you made.

💡 One paragraph, or even 2-3 sentences are also fine for describing an objective function.

Step 1: Announce the objective

Our objective has two terms: a classification loss on the labelled reports, and a consistency loss between the two views.

Step 1: Announce the objective

> Our objective has two terms: a classification loss on the labelled reports, and a consistency loss between the two views.



Say how many before you say what they are.
The reader then knows when you have finished.

Step 1: Announce the objective

> Our objective has two terms: a classification loss on the labelled reports, and a consistency loss between the two views.



Say how many before you say what they are.
The reader then knows when you have finished.

❌ "We use a combination of **several** losses."

✅ "Our objective has **two** terms."

Same rule as the baselines in Week 3: group and count before you list.

Step 2: One clause per term

💡 Every term answers one question: **what is it for?**

L_{cls} is the standard cross-entropy over the 14 findings.

L_{con} penalizes disagreement between the two views; it is what lets the model learn from the unlabelled reports.

⚠️ Without this step, the reader cannot tell which term carries your contribution.

Step 2: One clause per term (another example)

We estimate $Y_i = \beta_0 + \beta_1 X_i + \gamma' Z_i + \epsilon_i$, where X_i is the treatment of interest and Z_i is a vector of firm-level controls. We include firm size **because larger firms adopt earlier**. We also include industry fixed effects **because adoption rates differ sharply by sector**.

Step 2: One clause per term (another example)

the expression, one clause per term

the reason(s) for inclusion

We estimate $Y_i = \beta_0 + \beta_1 X_i + \gamma' Z_i + \epsilon_i$, where X_i is the treatment of interest and Z_i is a vector of firm-level controls. We include firm size **because larger firms adopt earlier**. We also include industry fixed effects **because adoption rates differ sharply by sector**.

Step 3: Justify the numbers you chose by hand

💡 Every constant that did not come from the data needs a sentence.

❌ $L = L_{\text{cls}} + 0.5 \cdot L_{\text{con}}$

✅ …where $\lambda = 0.5$ was selected on the validation set from $\{0.1, 0.5, 1.0\}$.

⚠️ The reader is asking one question: did you tune this on the test set?

Say where the number came from and the question goes away.

Three notation rules

- These are typography, not software. They apply in Word exactly as much.

  1. Function names upright, variables italic.

     - ❌ *Softmax*(*x*) ✅ Softmax(*x*)

  2. **One symbol, one job.** N cannot be both the number of samples and the number of layers. Rename one.

  3. Every symbol gets a size. "h" is not a definition. "$h \in \mathbb{R}^d$" is.

- Open your Week 5 symbol list. Nothing you add today may collide with it.

An equation is part of a sentence

The model minimizes:

$$L = L_{\text{cls}} + \lambda \cdot L_{\text{con}},$$

where λ controls the weight of the consistency term.

- the comma after the equation — the sentence has not ended
- the "where" line — every new symbol is named immediately after it

Example in MedCPT's paper

Jin, Qiao, et al. "Medcpt: Contrastive pre-trained transformers with large-scale pubmed search logs for zero-shot biomedical information retrieval." Bioinformatics 39.11 (2023): btad651.

~~queries for d_i~~. For the training instance i, we calculate its query-to-document loss $\mathcal{L}_i^{q2d}$ and document-to-query loss $\mathcal{L}_i^{d2q}$ by:

$$\mathcal{L}_i^{q2d} = -\log\left(\frac{\exp\left(E(q_i)^T E(d_i)\right)}{\sum_{m=1}^{|B|}\exp\left(E(q_i)^T E(d_m)\right)}\right) \text{ and } \mathcal{L}_i^{d2q} = -\log\left(\frac{\exp\left(E(q_i)^T E(d_i)\right)}{\sum_{m=1}^{|B|}\exp\left(E(q_m)^T E(d_i)\right)}\right)$$

We further weight the loss of instances by their clicks: $\mathcal{L}_B^{q2d} = \sum_{i=1}^{|B|} w_i \mathcal{L}_i^{q2d}$ and $\mathcal{L}_B^{d2q} = \sum_{i=1}^{|B|} w_i \mathcal{L}_i^{d2q}$, where $w_i = \frac{\log_2(c_i+1)}{\sum_{k=1}^{|B|}\log_2(c_k+1)}$. The final loss $\mathcal{L}_B$ of the mini-batch is their weighted sum and is optimized by gradient-based methods.

Your Turn

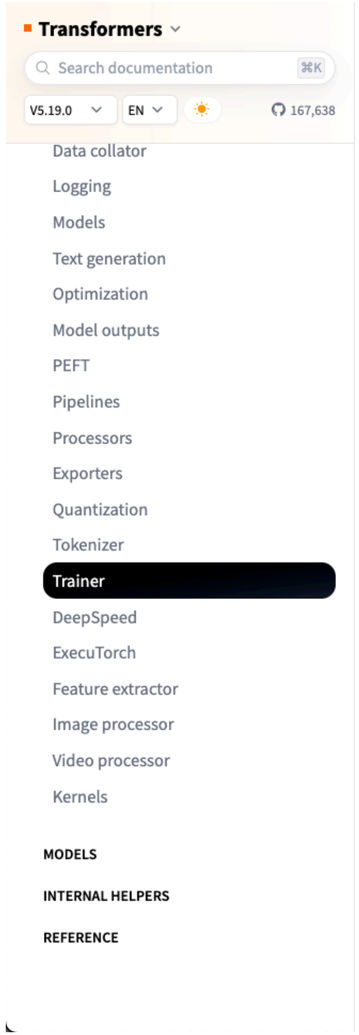
1. Announce — state the objective, and say how many parts it has.

2. Explain — one clause per term: what it is for.

3. Justify — every number and every choice you made.

Training Details

Why do we need to write Training Details?

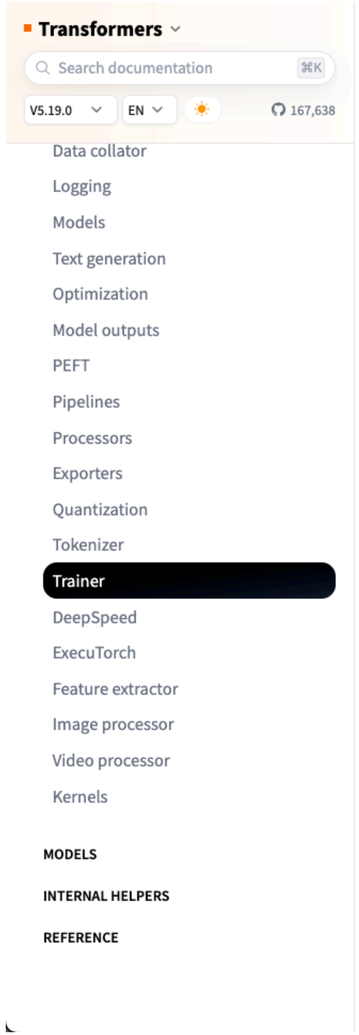


TrainingArguments

class transformers.TrainingArguments <source>

```
( output_dir: str | None = None, per_device_train_batch_size: int = 8, num_train_epochs: float = 3.0,
max_steps: int = -1, learning_rate: float = 5e-05, lr_scheduler_type:
transformers.trainer_utils.SchedulerType | str = 'linear', lr_scheduler_kwargs: dict | str | None =
None, warmup_steps: float = 0, optim: transformers.training_args.OptimizerNames | str =
'adamw_torch_fused', optim_args: str | None = None, weight_decay: float = 0.0, adam_beta1: float = 0.9,
adam_beta2: float = 0.999, adam_epsilon: float = 1e-08, optim_target_modules: None | str | list[str] =
None, gradient_accumulation_steps: int = 1, average_tokens_across_devices: bool = True, max_grad_norm:
float = 1.0, label_smoothing_factor: float = 0.0, bf16: bool = False, fp16: bool = False,
bf16_full_eval: bool = False, fp16_full_eval: bool = False, tf32: bool | None = None,
gradient_checkpointing: bool = False, gradient_checkpointing_kwargs: dict[str, typing.Any] | str | None
= None, torch_compile: bool = False, torch_compile_backend: str | None = None, torch_compile_mode: str |
None = None, use_liger_kernel: bool = False, liger_kernel_config: dict[str, bool] | None = None,
use_cache: bool = False, neftune_noise_alpha: float | None = None, torch_empty_cache_steps: int | None =
None, auto_find_batch_size: bool = False, logging_strategy: transformers.trainer_utils.IntervalStrategy
| str = 'steps', logging_steps: float = 500, logging_first_step: bool = False, log_on_each_node: bool =
True, logging_nan_inf_filter: bool = True, include_num_input_tokens_seen: str | bool = 'no', log_level:
str = 'passive', log_level_replica: str = 'warning', disable_tqdm: bool | None = None, report_to: None |
str | list[str] = 'none', run_name: str | None = None, project: str = 'huggingface', trackio_space_id:
str | None = None, trackio_bucket_id: str | None = None, trackio_static_space_id: typing.Union[str,
NoneType, typing.Literal[False]] = None, eval_strategy: transformers.trainer_utils.IntervalStrategy |
str = 'no', eval_steps: float | None = None, eval_delay: float = 0, per_device_eval_batch_size: int = 8,
prediction_loss_only: bool = False, eval_on_start: bool = False, eval_do_concat_batches: bool = True,
eval_use_gather_object: bool = False, eval_accumulation_steps: int | None = None, include_for_metrics:
list = <factory>, batch_eval_metrics: bool = False, save_only_model: bool = False, save_strategy:
transformers.trainer_utils.SaveStrategy | str = 'steps', save_steps: float = 500, save_on_each_node:
bool = False, save_total_limit: int | None = None, enable_jit_checkpoint: bool = False, push_to_hub:
bool = False, hub_token: str | None = None, hub_private_repo: bool | None = None, hub_model_id: str |
None = None, hub_strategy: transformers.trainer_utils.HubStrategy | str = 'every_save', hub_always_push:
bool = False, hub_revision: str | None = None, load_best_model_at_end: bool = False,
metric_for_best_model: str | None = None, greater_is_better: bool | None = None, ignore_data_skip: bool
= False, restore_callback_states_from_checkpoint: bool = False, full_determinism: bool = False, seed:
int = 42, data_seed: int | None = None, use_cpu: bool = False, accelerator_config: dict | str | None =
```

Why do we need to write Training Details?



TrainingArguments

class transformers.TrainingArguments <source>

```
( output_dir: str | None = None, per_device_train_batch_size: int = 8, num_train_epochs: float = 3.0,
max_steps: int = -1, learning_rate: float = 5e-05, lr_scheduler_type:
transformers.trainer_utils.SchedulerType | str = 'linear', lr_scheduler_kwargs: dict | str | None =
None, warmup_steps: float = 0, optim: transformers.training_args.OptimizerNames | str =
'adamw_torch_fused', optim_args: str | None = None, weight_decay: float = 0.0, adam_beta1: float = 0.9,
adam_beta2: float = 0.999, adam_epsilon: float = 1e-08, optim_target_modules: None | str | list[str] =
None, gradient_accumulation_steps: int = 1, average_tokens_across_devices: bool = True, max_grad_norm:
float = 1.0, label_smoothing_factor: float = 0.0, bf16: bool = False, fp16: bool = False,
bf16_full_eval: bool = False, fp16_full_eval: bool = False, tf32: bool | None = None,
gradient_checkpointing: bool = False, gradient_checkpointing_kwargs: dict[str, typing.Any] | str | None
= None, torch_compile: bool = False, torch_compile_backend: str | None = None, torch_compile_mode: str |
None = None, use_liger_kernel: bool = False, liger_kernel_config: dict[str, bool] | None = None,
use_cache: bool = False, neftune_noise_alpha: float | None = None, torch_empty_cache_steps: int | None =
None, auto_find_batch_size: bool = False, logging_strategy: transformers.trainer_utils.IntervalStrategy
| str = 'steps', logging_steps: float = 500, logging_first_step: bool = False, log_on_each_node: bool =
True, logging_nan_inf_filter: bool = True, include_num_input_tokens_seen: str | bool = 'no', log_level:
str = 'passive', log_level_replica: str = 'warning', disable_tqdm: bool | None = None, report_to: None |
str | list[str] = 'none', run_name: str | None = None, project: str = 'huggingface', trackio_space_id:
str | None = None, trackio_bucket_id: str | None = None, trackio_static_space_id: typing.Union[str,
NoneType, typing.Literal[False]] = None, eval_strategy: transformers.trainer_utils.IntervalStrategy |
str = 'no', eval_steps: float | None = None, eval_delay: float = 0, per_device_eval_batch_size: int = 8,
prediction_loss_only: bool = False, eval_on_start: bool = False, eval_do_concat_batches: bool = True,
eval_use_gather_object: bool = False, eval_accumulation_steps: int | None = None, include_for_metrics:
list = <factory>, batch_eval_metrics: bool = False, save_only_model: bool = False, save_strategy:
transformers.trainer_utils.SaveStrategy | str = 'steps', save_steps: float = 500, save_on_each_node:
bool = False, save_total_limit: int | None = None, enable_jit_checkpoint: bool = False, push_to_hub:
bool = False, hub_token: str | None = None, hub_private_repo: bool | None = None, hub_model_id: str |
None = None, hub_strategy: transformers.trainer_utils.HubStrategy | str = 'every_save', hub_always_push:
bool = False, hub_revision: str | None = None, load_best_model_at_end: bool = False,
metric_for_best_model: str | None = None, greater_is_better: bool | None = None, ignore_data_skip: bool
= False, restore_callback_states_from_checkpoint: bool = False, full_determinism: bool = False, seed:
int = 42, data_seed: int | None = None, use_cpu: bool = False, accelerator_config: dict | str | None =
```

A great number of settings to perform model training!

What you need to prepare (example)

learning rate	1E-04
batch size	64
number training epochs	10
warmup steps	1000
early stopping strategy	on validation F1
GPU	RTX 5090 * 1
Software versions	PyTorch 2.3, transformers 4.44

Example 1

Li, Hsiu-Wen, et al. "Improved unsupervised Chinese word segmentation using pre-trained knowledge and pseudo-labeling transfer." EMNLP 2023.

For the classifier, we use the pre-trained Chinese BERT-BASE model. We use Adam (Kingma and Ba, 2015) with learning rates of 5e-3 for the segment model and [1e-5, 5e-5] for the classifier. We train the segment model for 6,000 steps for each dataset, including the initial 800 steps for linear learning rate warm-up. The classifier is trained for 1,000 steps on each task with early stopping. More training details and training progress of our approach without using early stopping can be found at Appendix A.2 and A.3.

Example 2

Wang, Chien-Yao, I-Hau Yeh, and Hong-Yuan Mark Liao. "Yolov9: Learning what you want to learn using programmable gradient information." ECCV 2024.

The training parameters of YOLOv9 are shown in Table 1. We fully follow the settings of YOLOv7 AF [63], which is to use SGD optimizer to train 500 epochs. We first warm-up for 3 epochs and only update the bias during the warm-up stage. Next we step down from the initial learning rate 0.01 to 0.0001 in linear decay manner, and the data augmentation settings are listed in the bottom part of Table 1. We shut down mosaic data augmentation operations on the last 15 epochs.

Sentence Bank

- "Our objective has two terms."
- "L_x is the standard … over …"
- "L_y penalises …; it is what lets the model …"
- "…where $\lambda = 0.5$ was selected on the validation set from {…}."
- "We optimize X because our primary metric, Y, is not differentiable."
- "We select the final checkpoint by validation F1, not by validation loss."
- "All models were trained with AdamW (learning rate 2e-5) for 10 epochs."

Your Turn

NEXT WEEK ATTENTION!!

- Next week: Related Work (Literature Review).
- Please bring the following materials to class:
  - the 5-10 papers you would cite, with the Zotero entries already made
  - for each one, one sentence on how it differs from what you are doing
  - the two or three papers you consider closest to your own work

Not finished reading them?

- Bring the list anyway.
- Next week is about how to group them, not how many you have.

Thank you for your participation 📚