Understanding TrainingArguments

- per_device_train_batch_size:
  This parameter defines the number of samples processed on each GPU during a single training step, directly impacting both the speed of training and the memory usage of the GPU(carte graphique). If the desired batch size exceeds the GPU's memory capacity, techniques like gradient accumulation can be employed to manage memory effectively.

Batch Size	Training Time per Epoch	Model Performance
Small	Longer	Varies
Large	Shorter	Generally better

→ A step/iteration: In each step the model compute prediction with a forward pass to the next layer, then measure the difference between predictions and actual labels, and finally does a backward pass to updates model weights using backpropagation. COUNT HOW MANY TIME YOU UPDATE MODEL PARAMETERS

→ A batch: number of sample/token processed in one forward and backward pass.

→gradient_accumulation_steps: Instead of updating the model every step, we update every n numbers of step. Helpful when the batch size is too large to fit in GPU memory.

- fp16: help speed training and save GPU memory
- logging_steps: print the logs every n steps, For small datasets: Use smaller values (e.g., 5-10 steps) to monitor changes closely. For large datasets**:** Use larger values (e.g., 50-100 steps) to avoid excessive logging.

→ logs: record of metrics that are printing during training to monitor model's progress, it provides real time update loss, step, learning rate…

- save_steps: determines how often (n steps) the models is saved during training, so if it crash you can resume training at the last saved checkpoint
- eval_steps how often (n steps) the model is evaluated on the validation set, it checks metrics like accuracy,loss f1-score.
  For small datasets: Use smaller values (e.g., 5-10 steps) to monitor changes closely. For large datasets: Use larger values (e.g., 50-100 steps) to avoid excessive logging. For fast evaluation lower the number and for slow evaluation increase the number.
- weigth_decay: prevents large weights to improve generalization, help reduces overftting.

	recommended weigth_decay
small dataset	0.01-0.1
large dataset	0.001-0.01

fine-tuning transformers models	0.01 (default in Hugging Face)

→ loss: how far a model's predictions are from the real values, it tells how badly it perform during training. The loss function calculate the difference error between prediction and true value.

- load_best_model_at_end: during the training the model is evaluated at the end of each evaluation (steps or epoch), this parameters will automatically reload the checkpoint that achieved the best score on the test set and not just the final epoch model.
- per_device_eval_batch_size: same as per_device_train_batch_size except that it is the number of sample processed during the evaluation.
- learning_rate: Controls how much the model's weights are adjusted for each step during the training. It determines the size of the step the models take to reduce the loss during each iteration.
  → large learning rate (2e-4) are faster but riskier
  → small learning rate (1e-5,1e-3) are better for fine-tuning but slower
- warmup_steps: number of steps at the beginning of training, the learning rate start from a low value then gradually increases to the learning rate value set in the training arguments.

→ gradient: indicates how to change the weights to minimize the loss

- gradient_accumulation_steps: Instead of updating the weights after every batch, it does the gradient after n batches. It simulates a larger batch size, for example: train_batch= 6 and gradient_accumulation= 32, it will simulates a number of batch = 192 (32*6)