Convolution | Tokenizatio | image-2-10 | How GPT3 | 1706.0376 | Build a Larg | Build a Larg | Build a Larg | Build a Larg

miro.com/app/board/uXjVKENK1Qc=/

miro | Convolutional Neural Networks | Upgrade | Present | Share

tasks that the model wasn't explicitly trained to perform

Lecture6: Stages of building LLM → Our plan for this playlist!

AGE 1

STAGE 2

STAG

Attention mechanism

LLM architecture

Training loop

Model evaluation

Load pretrained weights

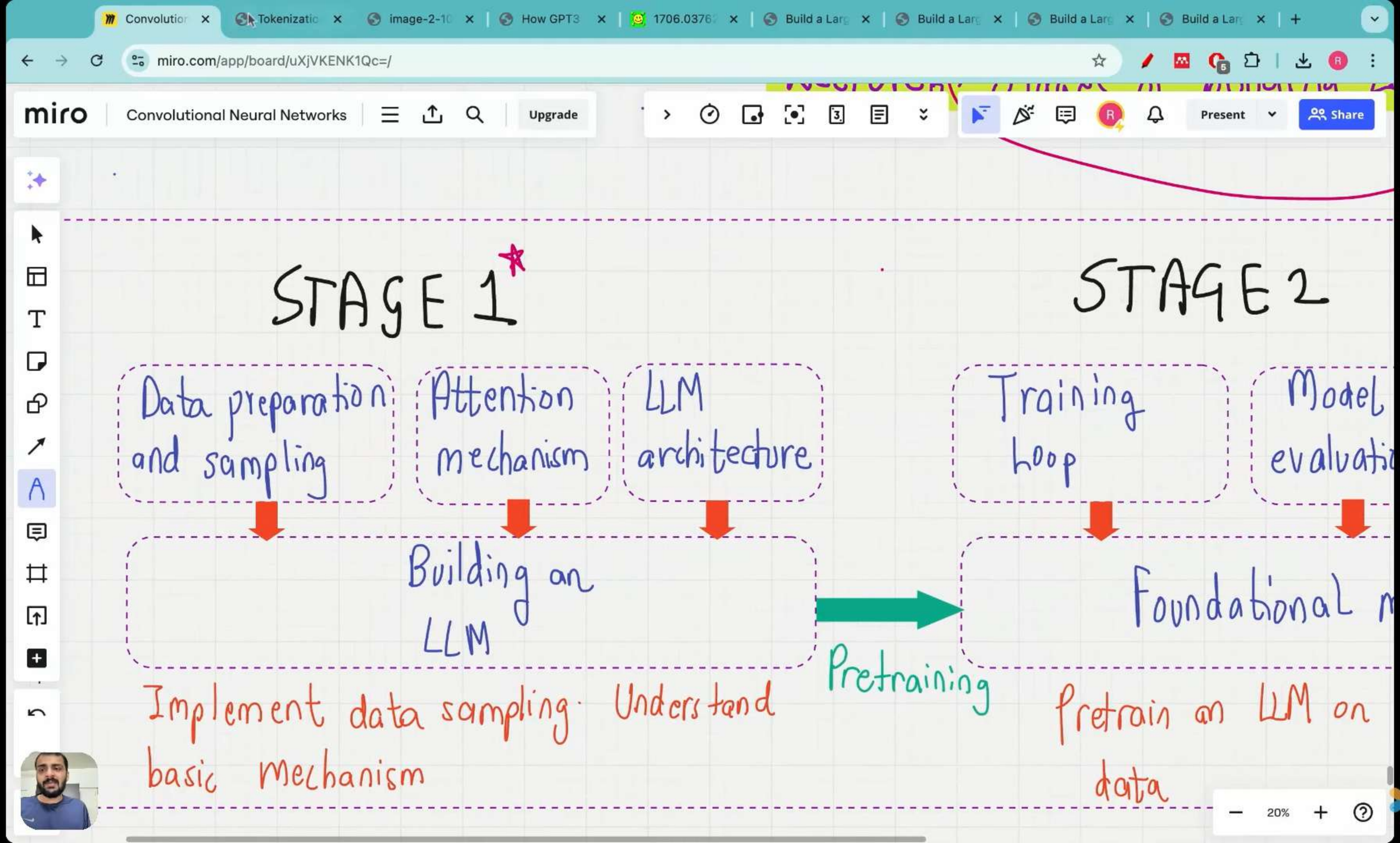
Finetuning

Classifier

Building an LLM

Foundational model

16%


Convolution
Tokenizatic
image-2-10
How GPT3
1706.0376
Build a Larg
Build a Larg
Build a Larg
Build a Larg
miro.com/app/board/uXjVKENK1Qc=/
miro
Convolutional Neural Networks
Upgrade
Present
Share
STAGE 1
STAGE 2
Data preparation and sampling
Attention mechanism
LLM architecture
Training loop
Model evaluati
Building an LLM
Foundational m
Pretraining
Implement data sampling. Understand basic mechanism
Pretrain an LLM on data
20%

Lecture6: Stages of building LLM

→ Our plan for this playlist!

STAGE 2

STAGE 3

LLM architecture

Training Loop

Model evaluation

Load pretrained weights

Foundational model

Pretraining

Pretrain an LLM on unlabeled data

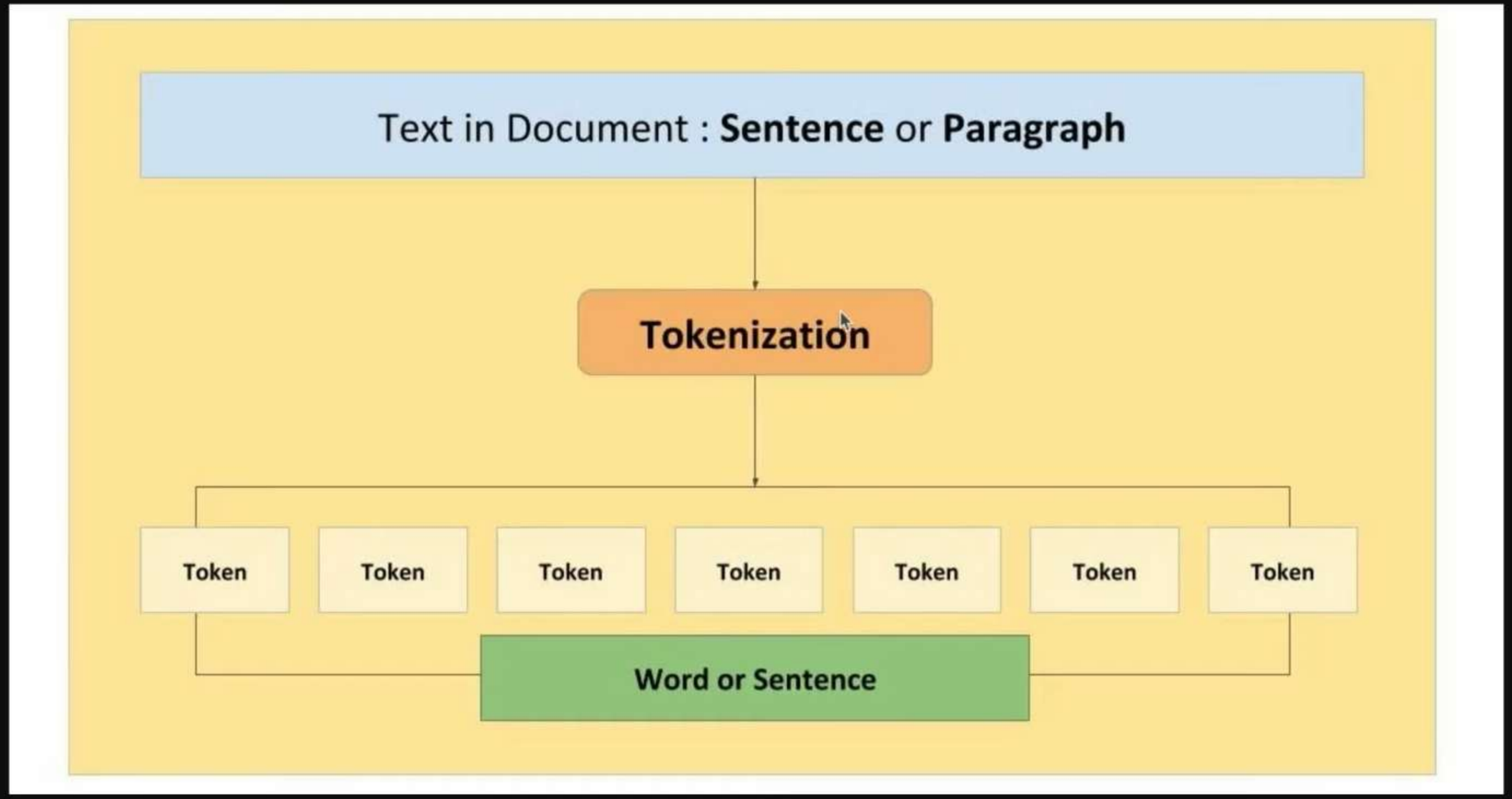
Finetuning

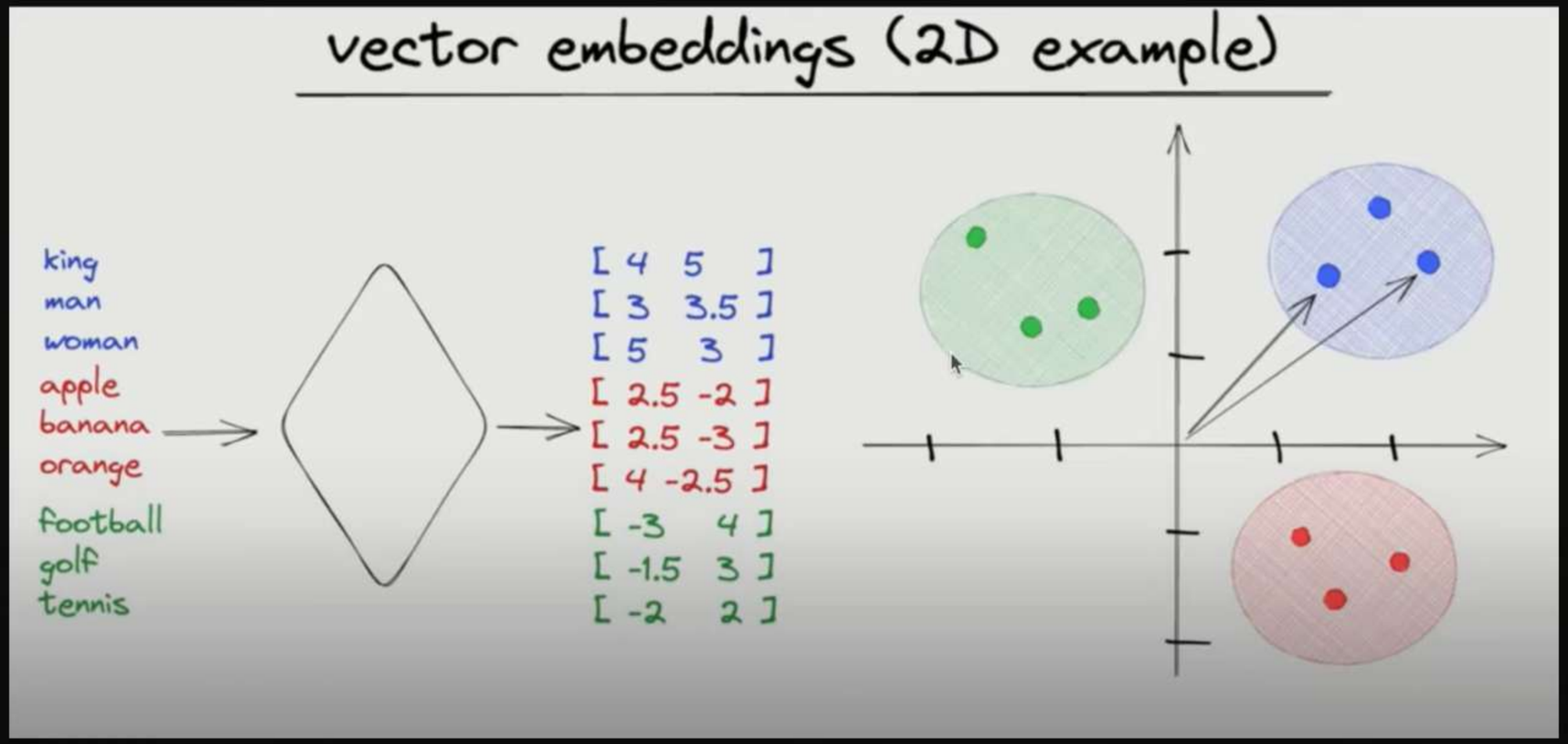
Classifier

Personal Assistant

Finetuning

Finetune the pretrained LLM

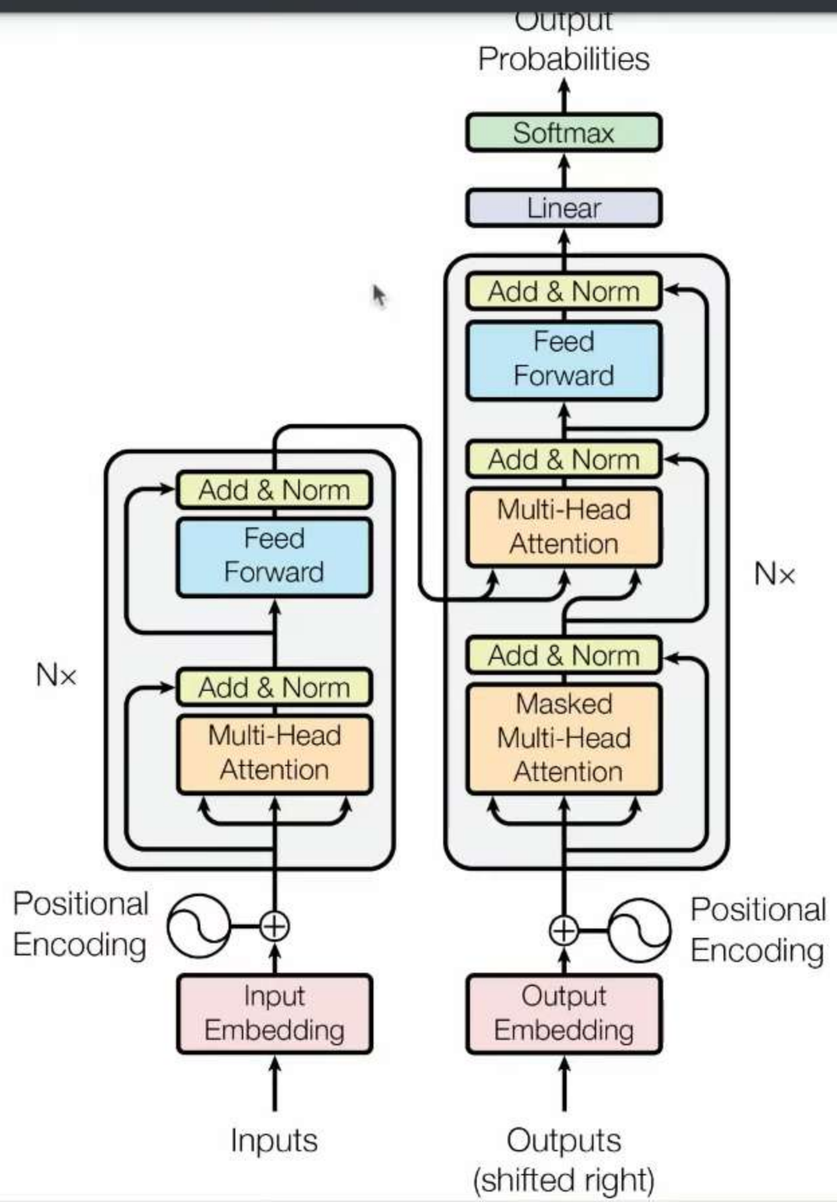

Text in Document : Sentence or Paragraph
Tokenization
Token
Token
Token
Token
Token
Token
Token
Word or Sentence

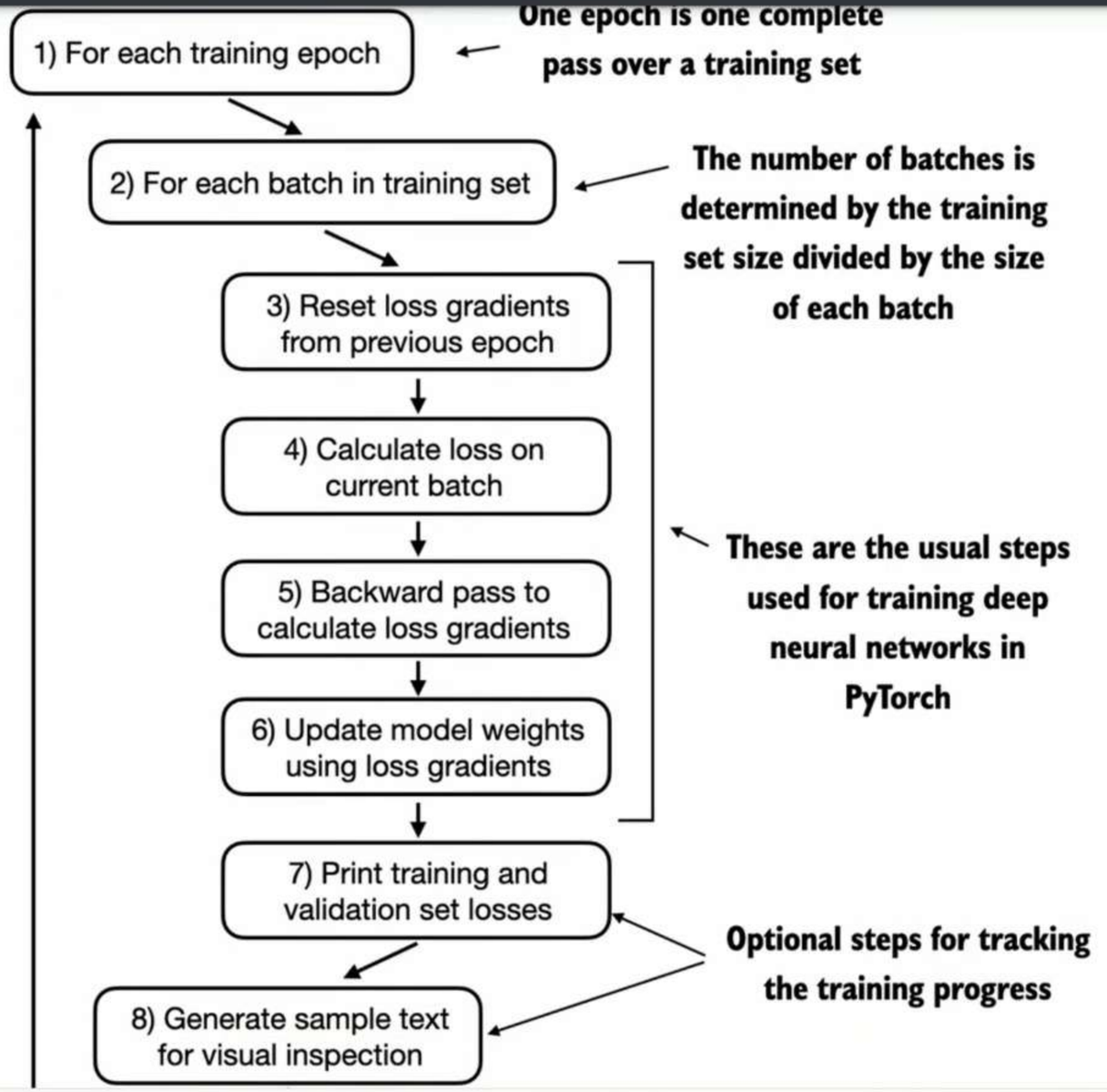


Generated training examples

xample #	Input (features)	Correct output (labels
1	Second law of robotics :	a
2	Second law of robotics : a	robot
3	Second law of robotics : a robot	must

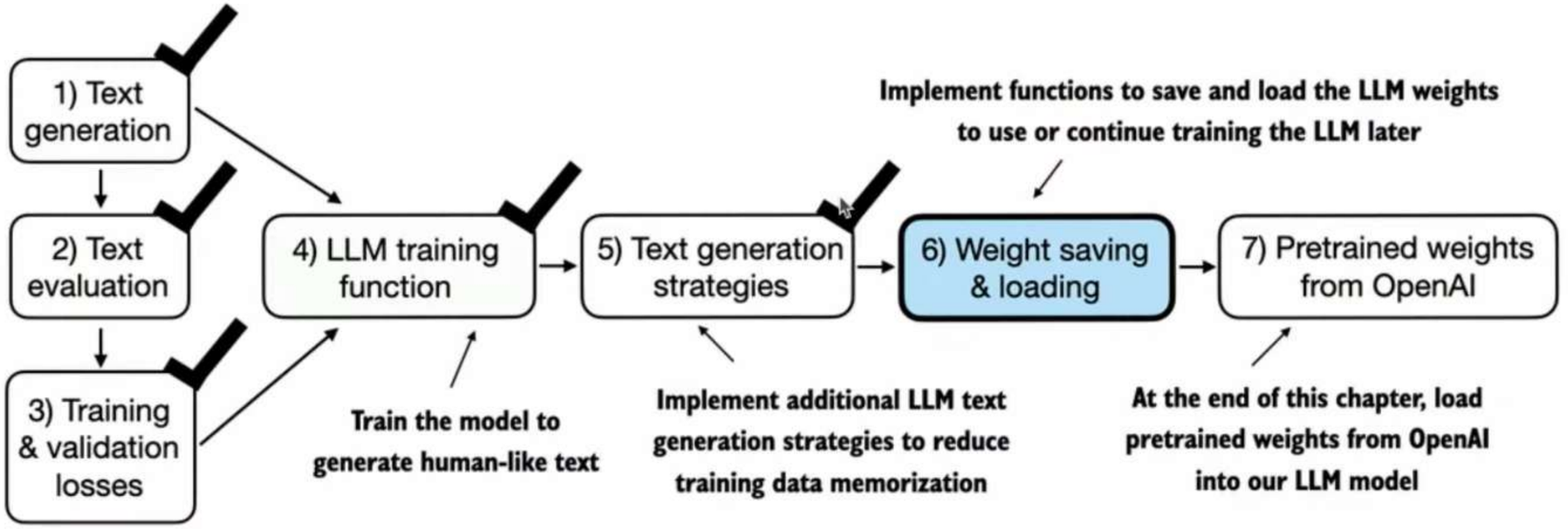
odel is presented with an example. We only show it the features and ask it to predict the next word


Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (shifted right)


1) For each training epoch
One epoch is one complete pass over a training set
2) For each batch in training set
The number of batches is determined by the training set size divided by the size of each batch
3) Reset loss gradients from previous epoch
4) Calculate loss on current batch
5) Backward pass to calculate loss gradients
6) Update model weights using loss gradients
These are the usual steps used for training deep neural networks in PyTorch
7) Print training and validation set losses
8) Generate sample text for visual inspection
Optional steps for tracking the training progress

e want to use it in a new session.

As illustrated in the chapter overview in Figure 5.16, we cover how to save and load etrained model in this section. Then, in the upcoming section, we will load a mor pable pretrained GPT model from OpenAI into our `GPTModel` instance.



gure 5.16 After training and inspecting the model, it is often helpful to save the model so that we can use ntinue training it later, which is the topic of this section before we load the pretrained model weights from enAI in the final section of this chapter.

plants from images, categorizing news articles into topics like sports, politics, or technology, and distinguishing between benign and malignant tumors in medical imaging.

The key point is that a classification-finetuned model is restricted to predicting classes it has encountered during its training—for instance, it can determine whether something is "spam" or "not spam," as illustrated in figure 6.3, but it can't say anything else about the input text.

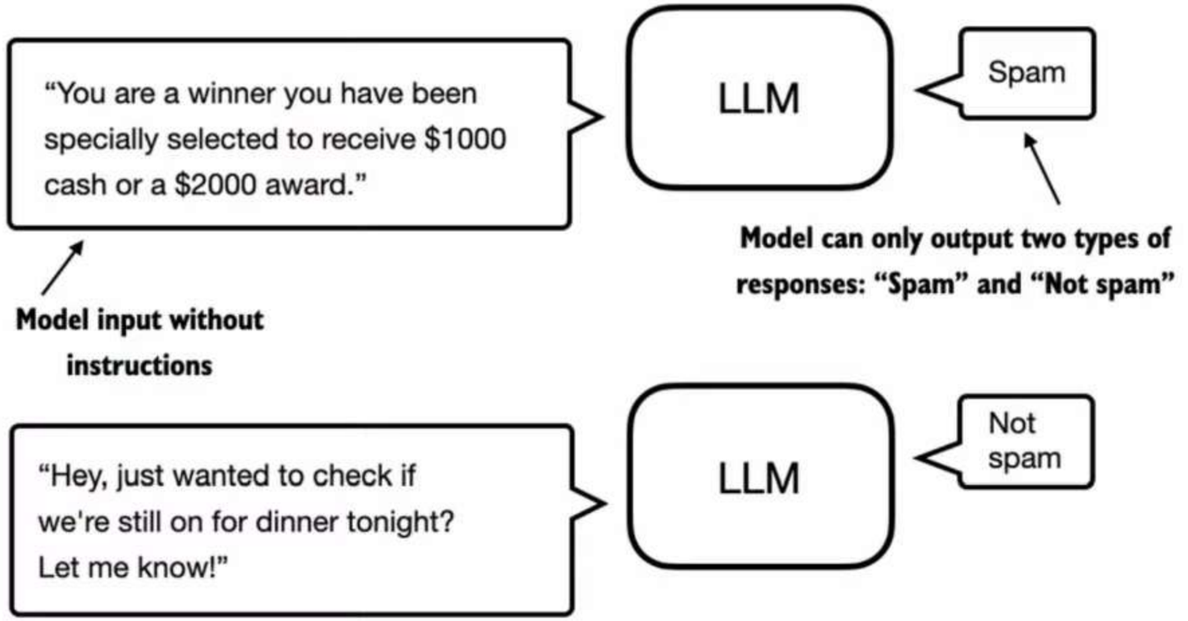
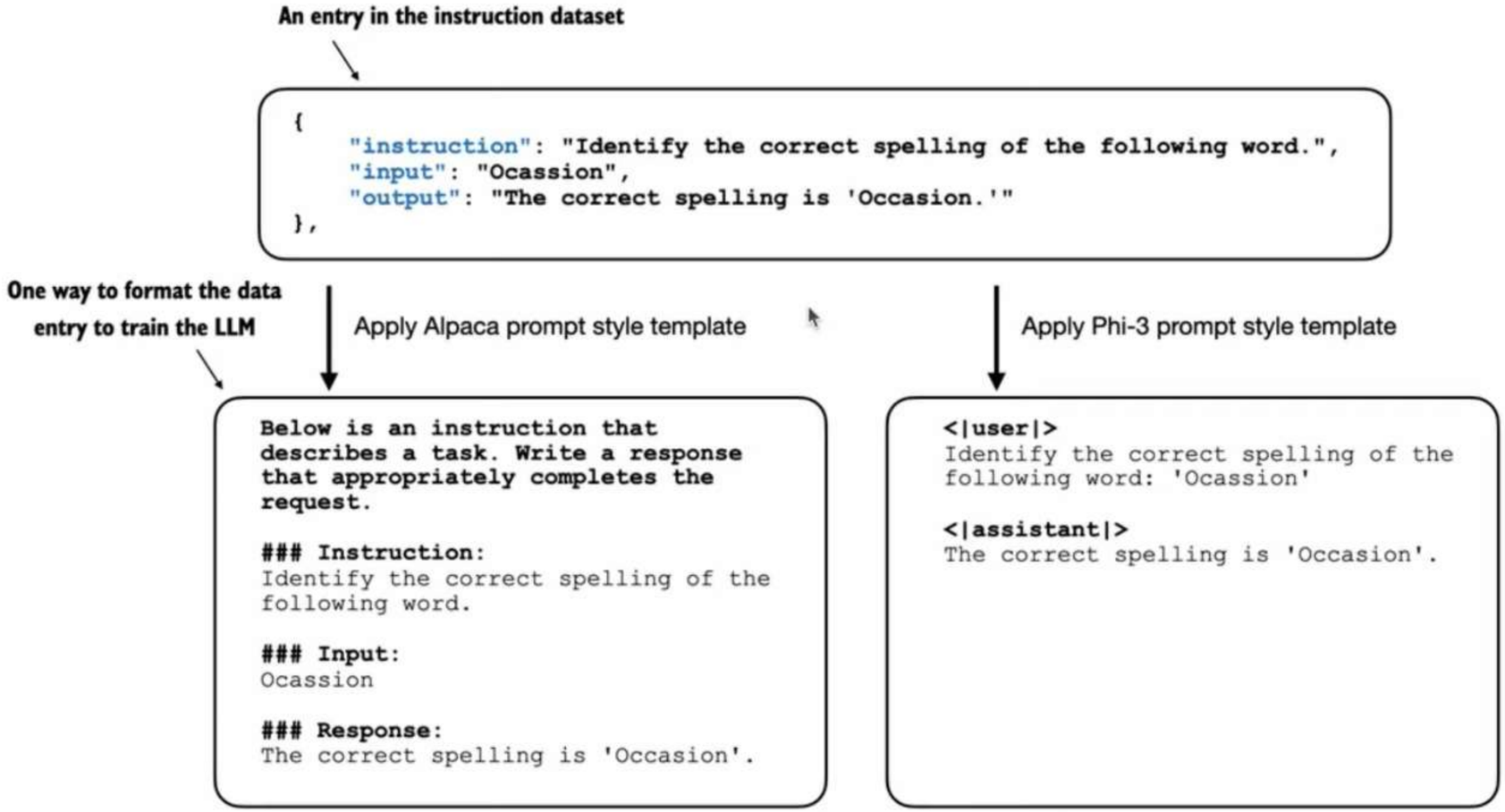


Figure 6.3 Illustration of a text classification scenario using an LLM. A model finetuned for spam classification does not require further instruction alongside the input. In contrast to an instruction-finetuned model, it can only respond with "spam" and "not spam."

In contrast to the classification-finetuned model depicted in figure 6.3, an instruction-finetuned model typically has the capability to undertake a broader range of tasks. We can

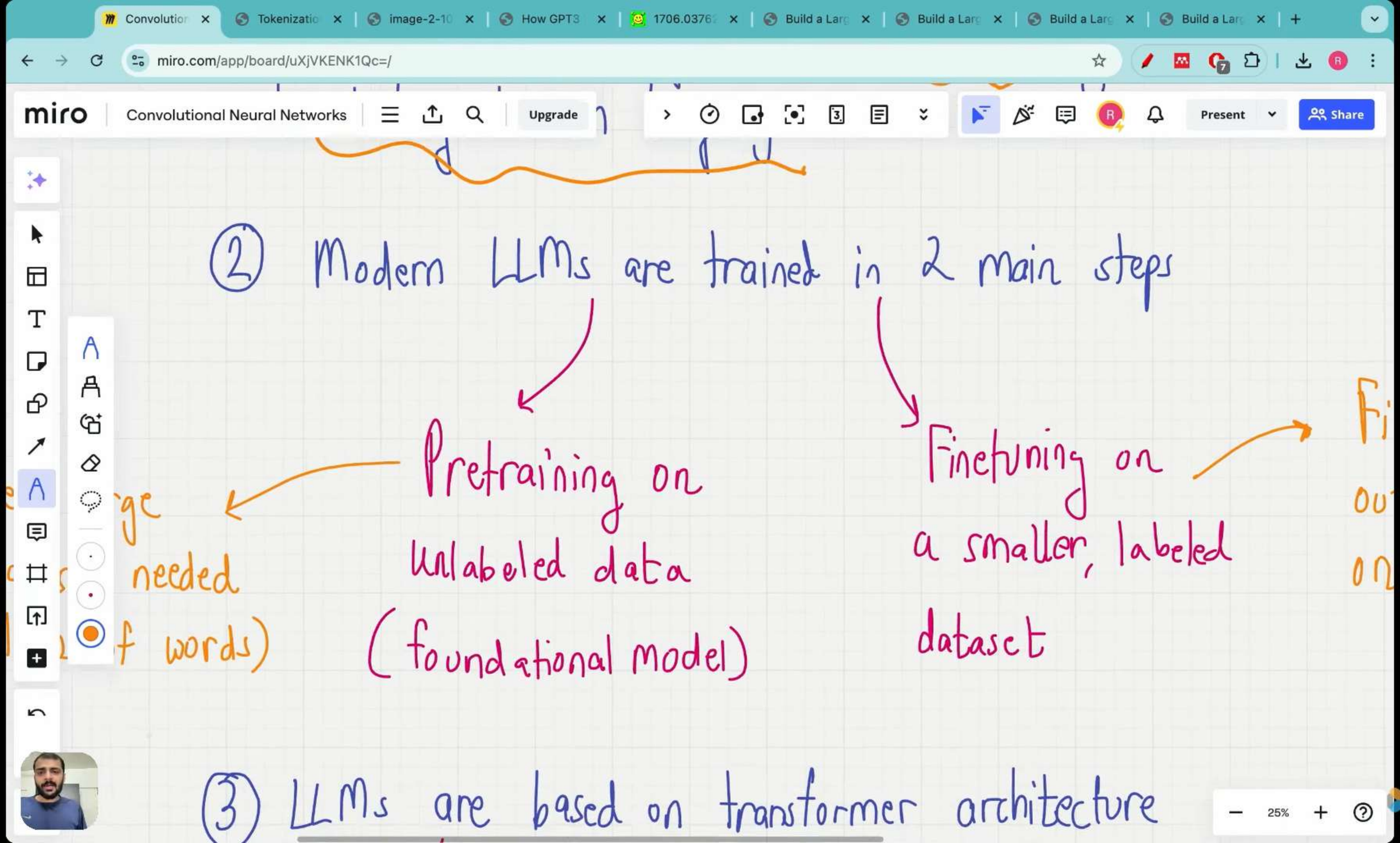
used in the training of notable LLMs such as Alpaca and Phi-3. Alpaca was one of the ear LLMs to publicly detail its instruction finetuning process. Phi-3, developed by Microsoft, included to demonstrate the diversity in prompt styles.

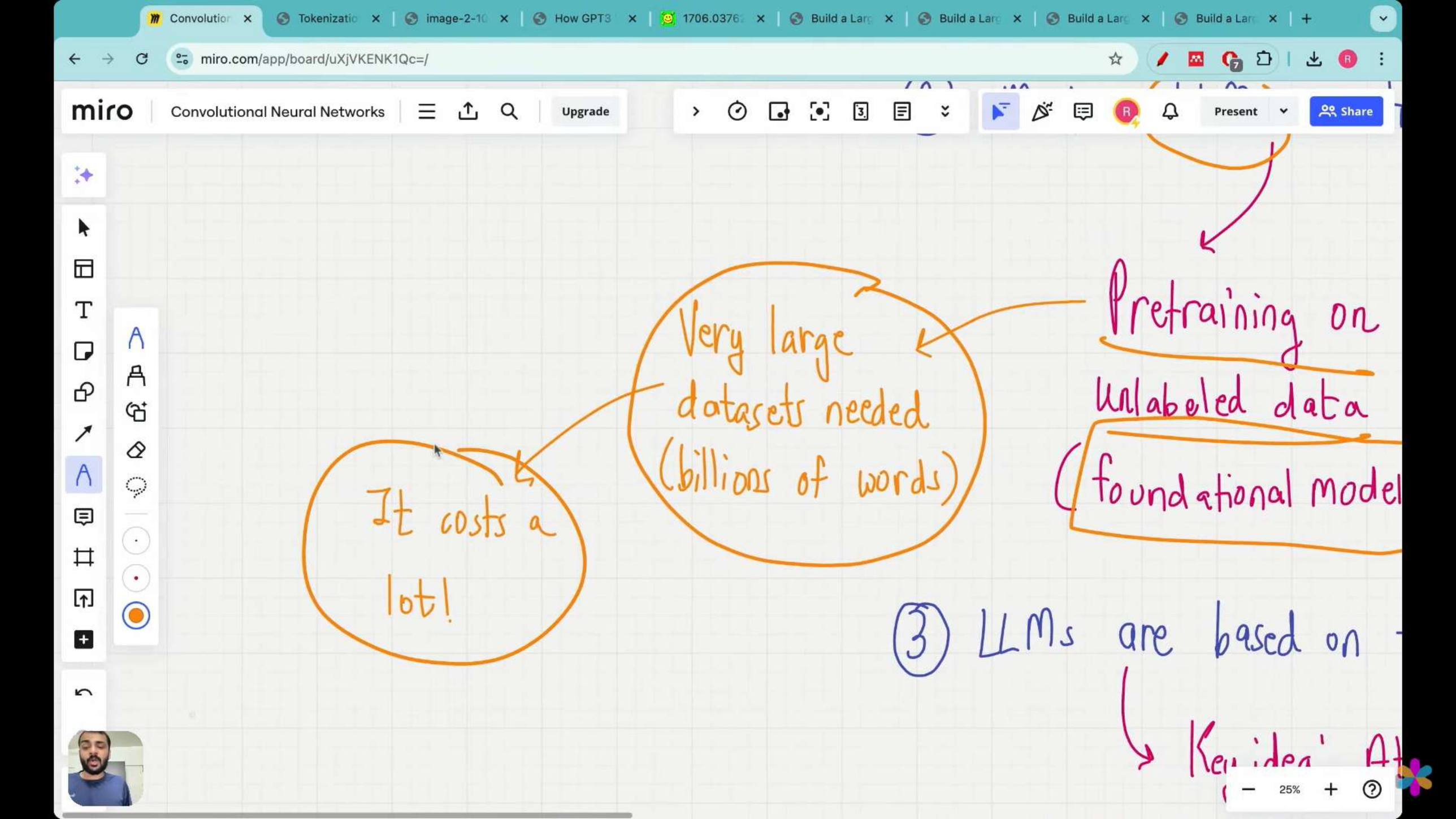


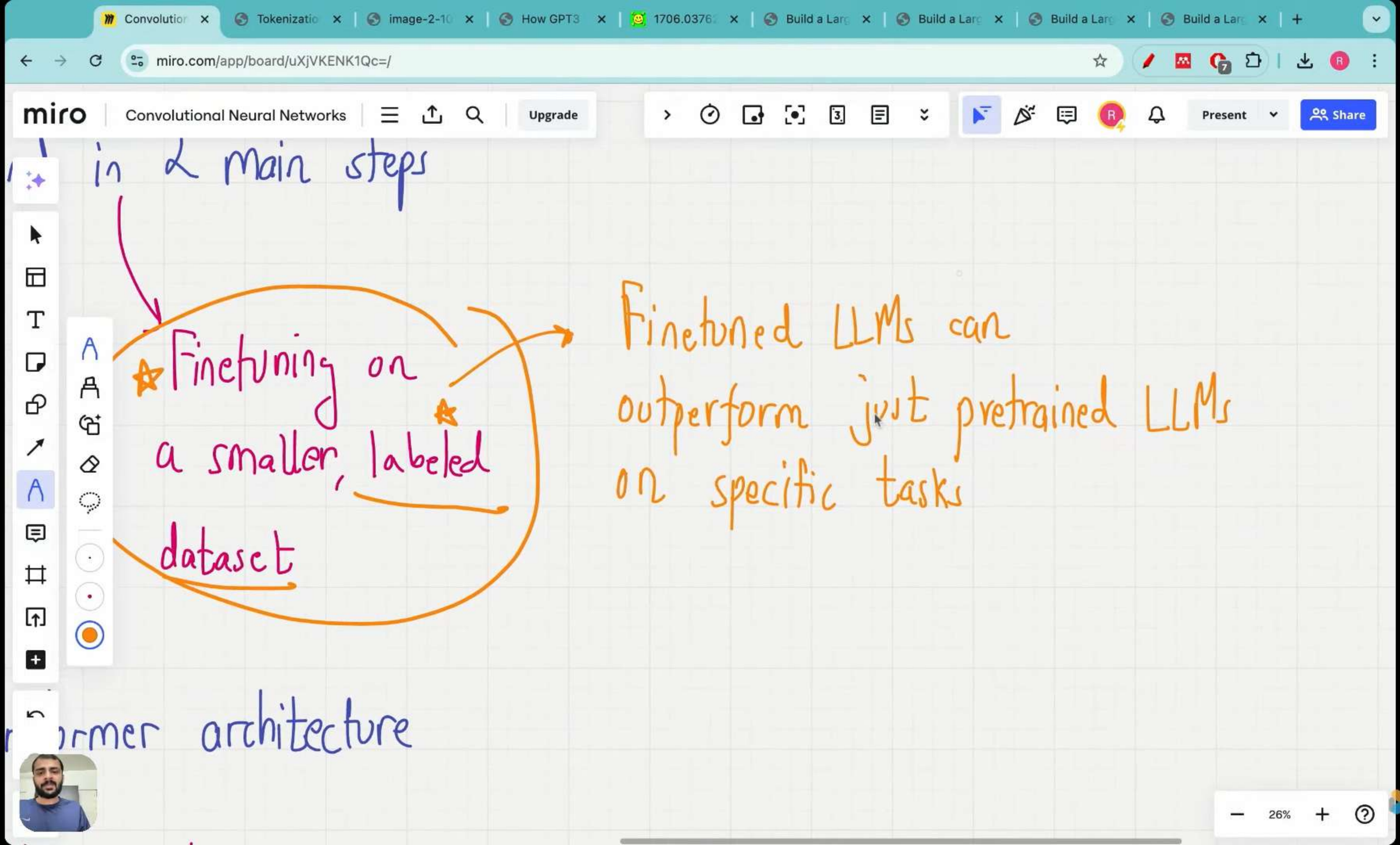
RECAP of what all we've learnt so far

① LLMs have transformed the field of NLP. They have led to advancements in generating, understanding and translating human language

② Modern LLMs are trained in 2 main steps


Convolution
Tokenizatio
image-2-10
How GPT3
1706.03762
Build a Larg
Build a Larg
Build a Larg
Build a Larg
miro.com/app/board/uXjVKENK1Qc=/
miro
Convolutional Neural Networks
Upgrade
Present
Share
2 Modern LLMs are trained in 2 main steps
Pretraining on
unlabeled data
(foundational model)
Finetuning on
a smaller, labeled
dataset
needed
f words)
3 LLMs are based on transformer architecture
25%


Convolution
Tokenizatio
image-2-10
How GPT3
1706.0376
Build a Larg
Build a Larg
Build a Larg
Build a Larg
miro.com/app/board/uXjVKENK1Qc=/
miro
Convolutional Neural Networks
Upgrade
Present
Share
Pretraining on
Unlabeled data
foundational model
Very large
datasets needed
(billions of words)
It costs a
lot!
3 LLMs are based on
25%


Convolution
Tokenizatio
image-2-10
How GPT3
1706.0376
Build a Larg
Build a Larg
Build a Larg
Build a Larg
miro.com/app/board/uXjVKENK1Qc=/
miro
Convolutional Neural Networks
Upgrade
Present
Share
in 2 main steps
Finetuning on a smaller, labeled dataset
Finetuned LLMs can outperform just pretrained LLMs on specific tasks
rmer architecture
26%

(3) LLMs are based on transformer architecture

↳ Key idea: Attention mechanism gives LLM selective access to whole input sequence when generating output one word at a time

(4) Original transformer: Encoder + Decoder

(4) Original transformer: Encoder + Decoder

(5) GPT: only Decoder, no Encoder

(6) While LLMs are only trained for predicting next word, they show emergent properties

→ ability to classify, translate and summarize texts

Lecture 7: Working with Text Data