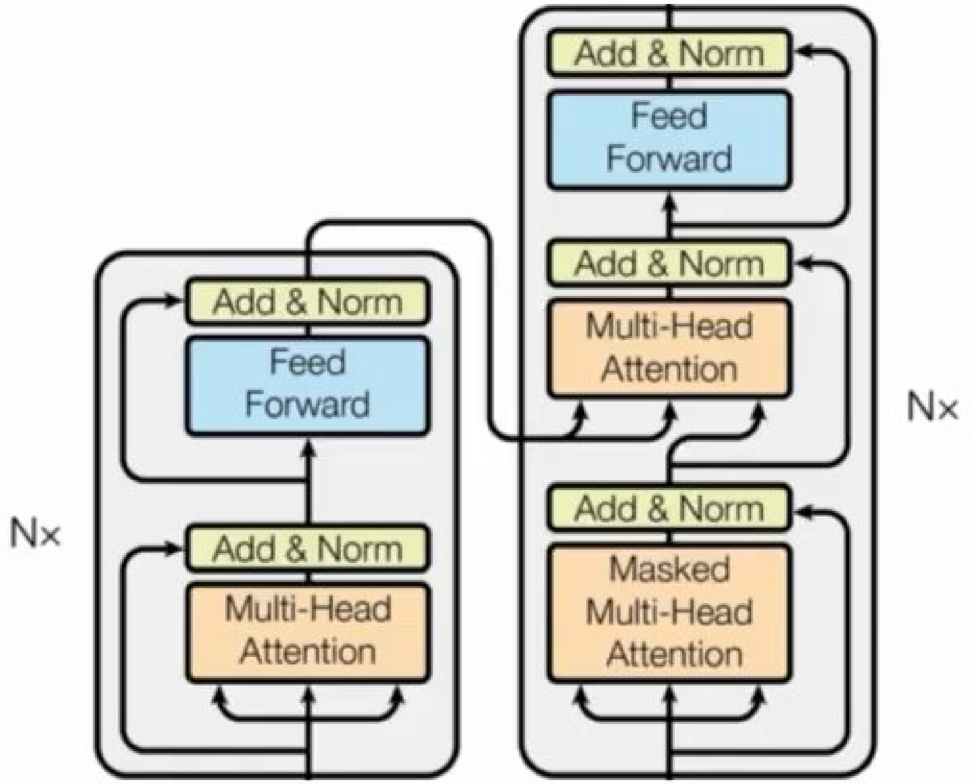
Stanford | ENGINEERING


Lecture 3
CME 295: Transformers & Large Language Models
Afshine Amidi & Shervine Amidi
Stanford University
ICME
Stanford

Recap of last episode...

3 categories of Transformer-based models:

Encoder-decoder	Text to text	T5, mT5, ByT5



Stanford

Figure adapted from "Attention is All You Need", Vaswani et al., 2017.

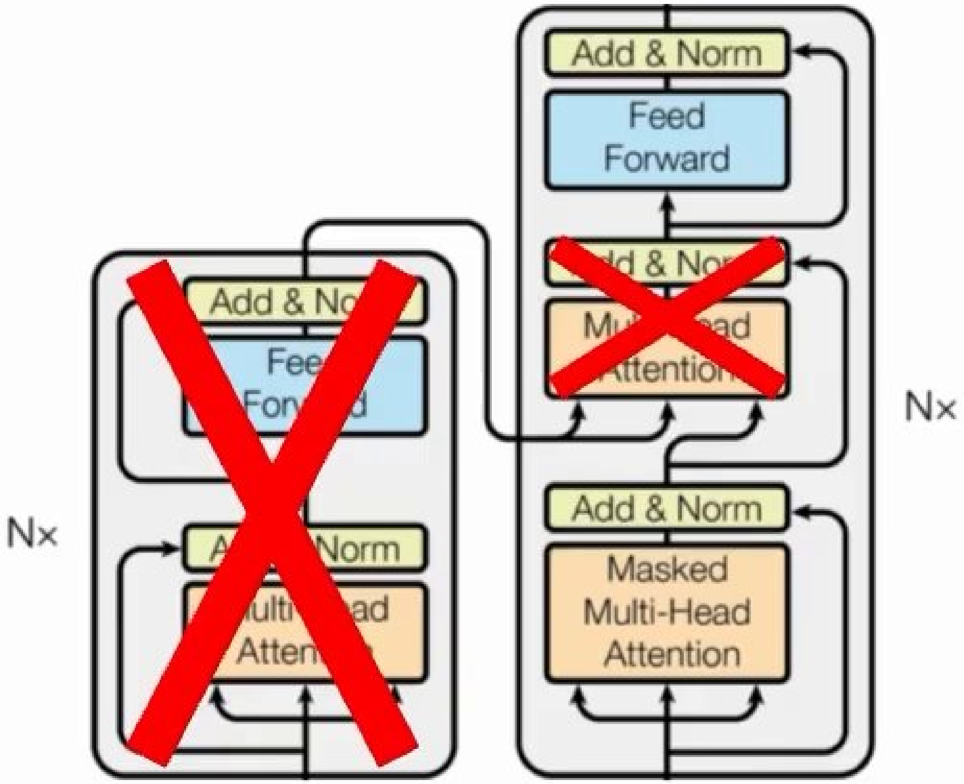

Recap of last episode...
3 categories of Transformer-based models:
Encoder-only
Projection of embedding for class prediction (e.g. sentiment extraction)
BERT, DistilBERT, RoBERTa
Stanford University
Stanford


Recap of last episode...
3 categories of Transformer-based models:
Encoder-only
Projection of embedding for class prediction (e.g. sentiment extraction)
BERT, DistilBERT, RoBERTa
Stanford University
ICME
Stanford

Recap of last episode...

3 categories of Transformer-based models:

Encoder-decoder	Text to text	T5, mT5, ByT5
Encoder-only	Projection of embedding for class prediction (e.g. sentiment extraction)	BERT, DistilBERT, RoBERTa
Decoder-only	Text to text	GPT series



Stanford

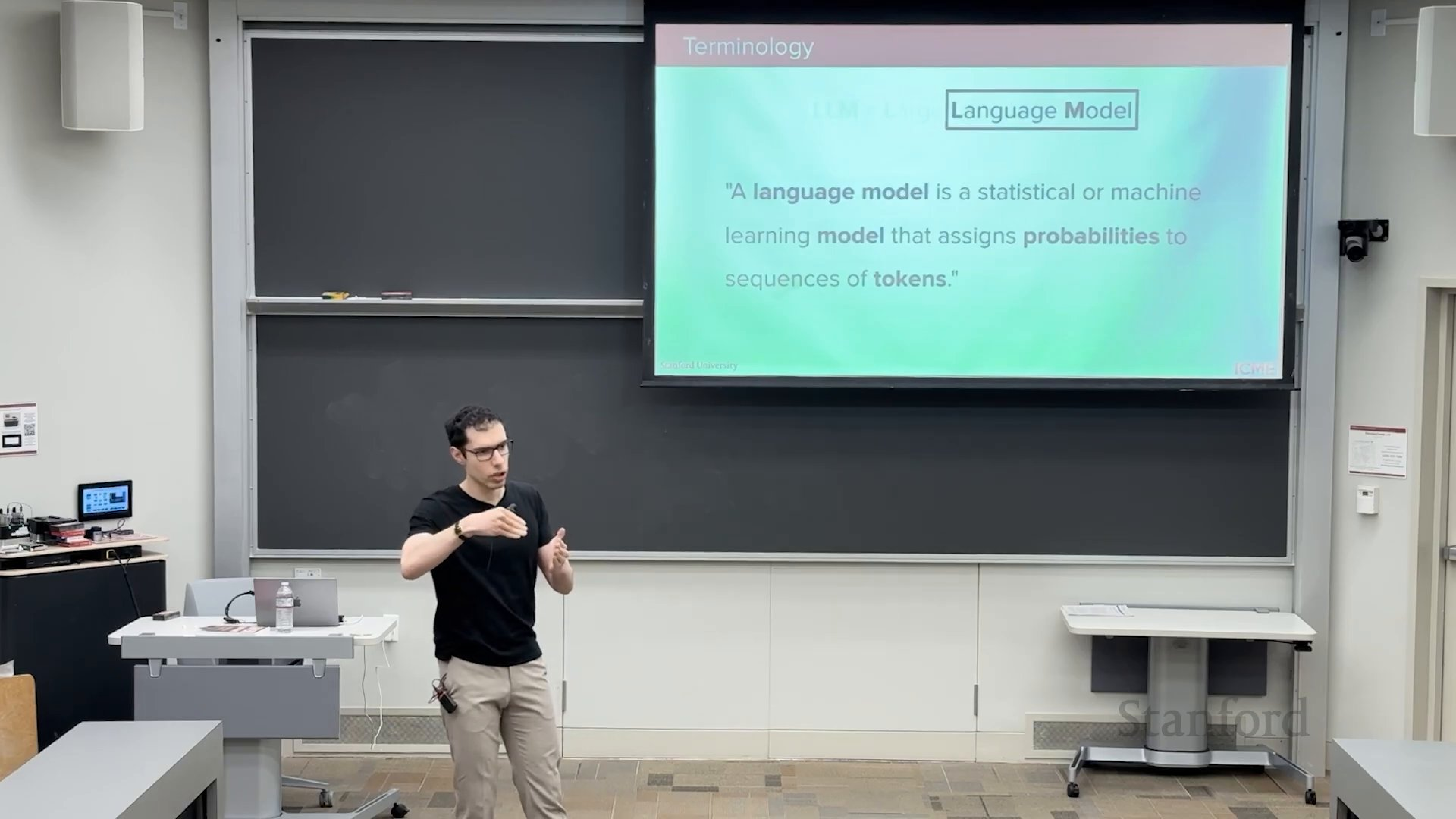
Recap of last episode...

3 categories of Transformer-based models:

Encoder-decoder	Text to text	T5, mT5, ByT5
Encoder-only	Projection of embedding for class prediction (e.g. sentiment extraction)	BERT, DistilBERT, RoBERTa
Decoder-only	Text to text	GPT series

Terminology

LLM = **L**arge **L**anguage **M**odel


Terminology
Language Model
"A language model is a statistical or machine learning model that assigns probabilities to sequences of tokens."
Stanford University
ICME
Stanford

Terminology

LLM = $\boxed{\text{Large}}$ Language Model

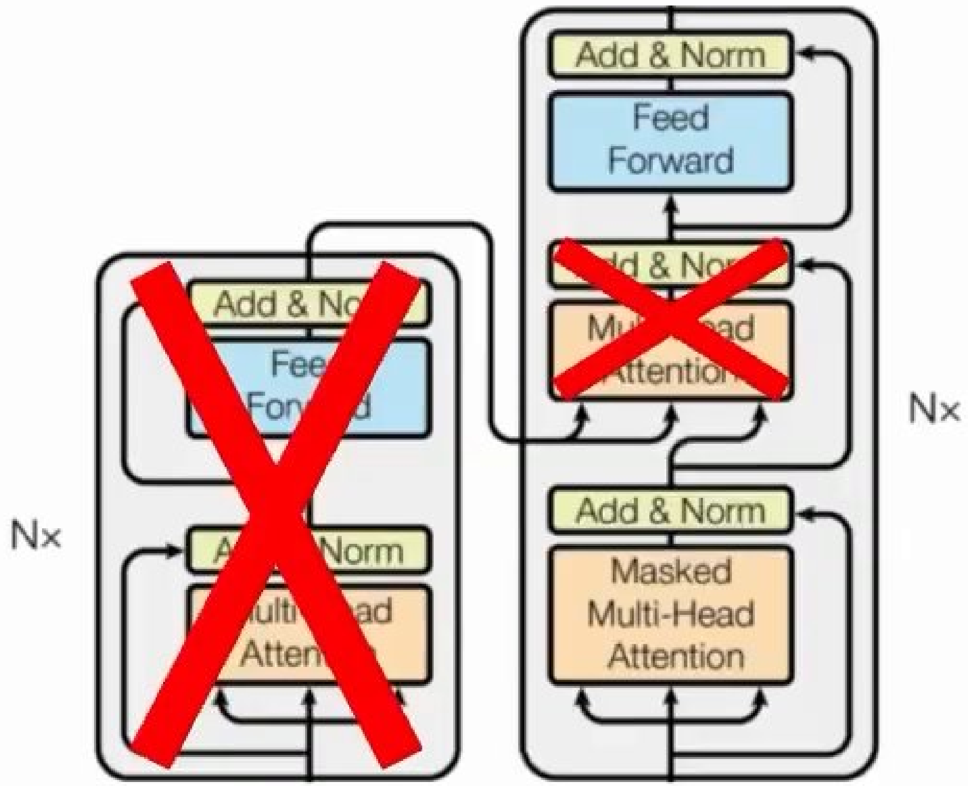
Stanford


Terminology
Large
Model size: billions of parameters or more
Training data: 100s of billions of tokens or more
Compute: a lot of GPUs
Stanford University
Stanford


Terminology
Large
Model size: billions of parameters or more
Training data: 100s of billions of tokens or more
Compute: a lot of GPUs
Stanford University
Stanford

Characteristics

Decoder-only **Transformer**-based model

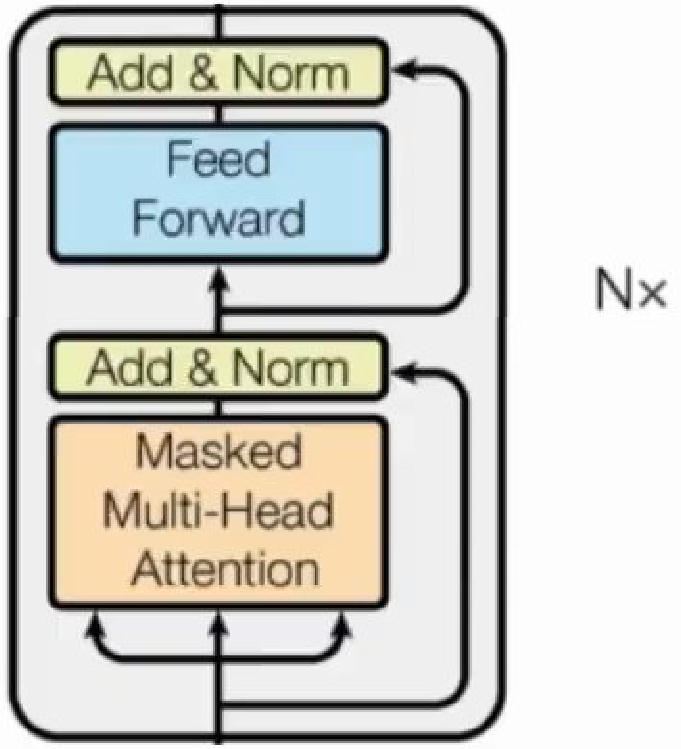


Stanford


Characteristics
Decoder-only **Transformer**-based model
Stanford University
ICME
Stanford

Characteristics

Decoder-only **Transformer**-based model



Stanford

Figure adapted from "Attention is All You Need", Vaswani et al., 2017.

Transformers & Large Language Models

LLM overview

MoE-based LLMs

Response generation

Prompting strategies

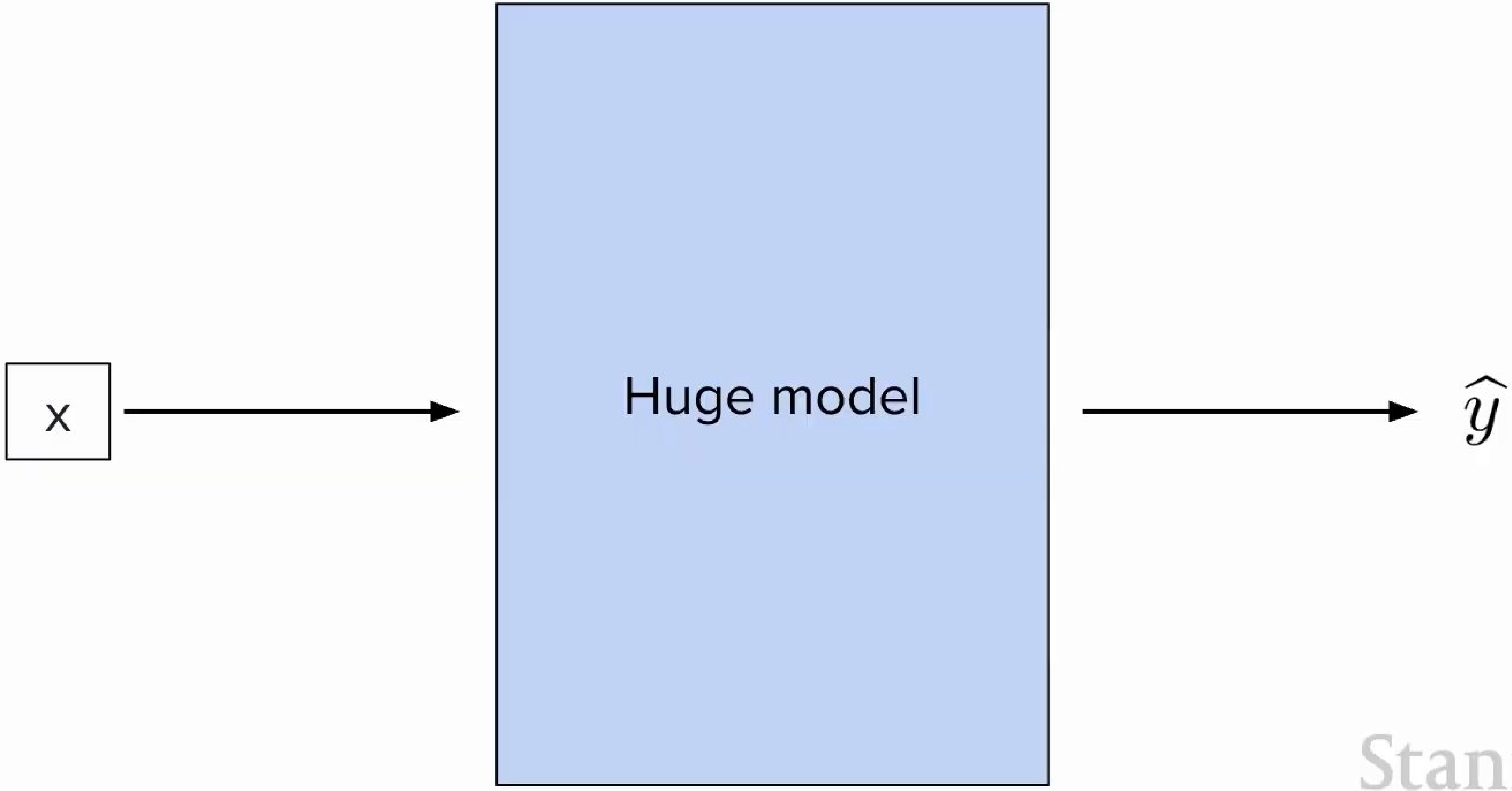
Inference optimizations

Stanford

Stanford University

ICME

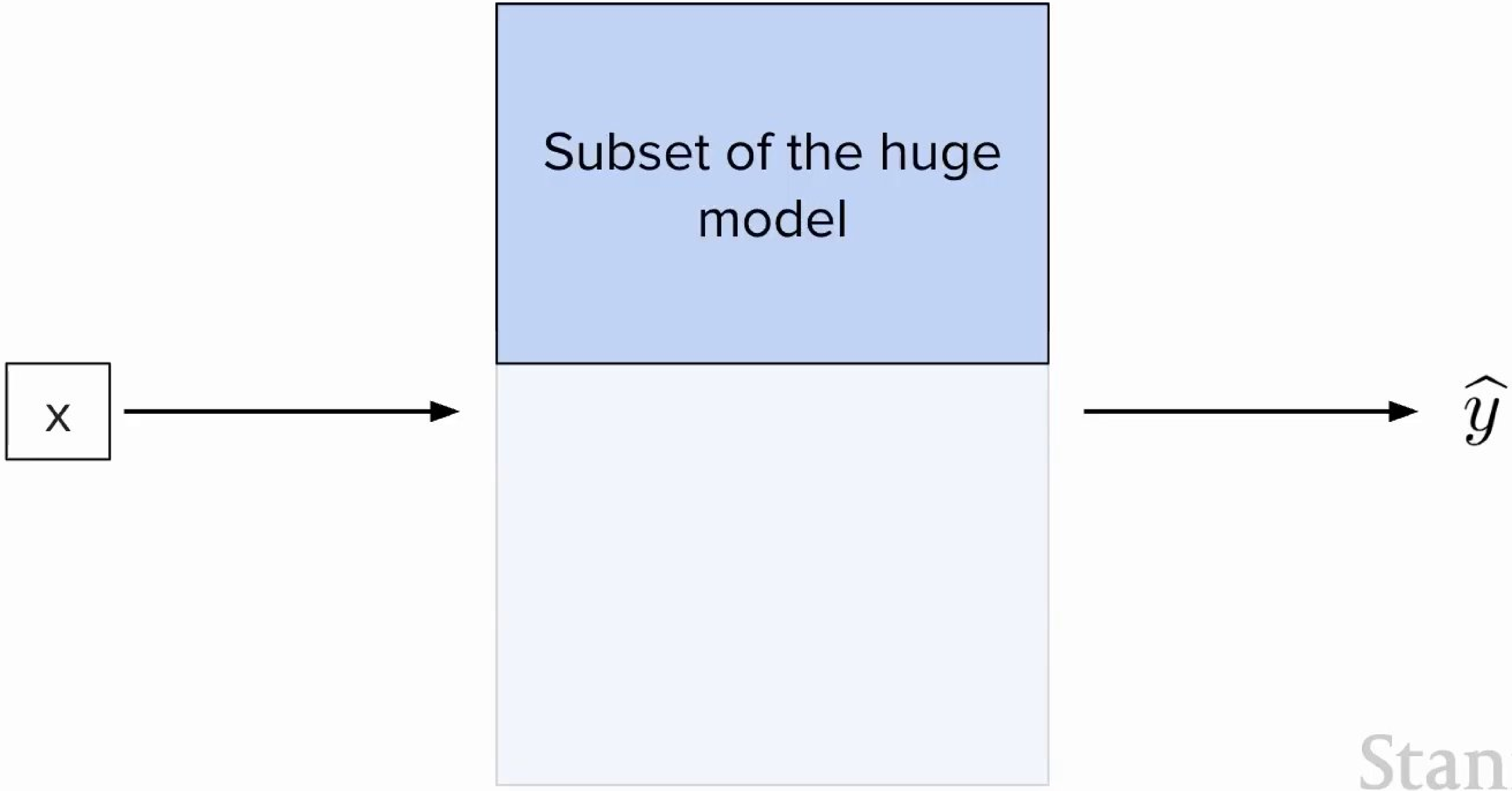
Motivation




Motivation
x
Huge model
$\hat{y}$
Stanford University
Stanford

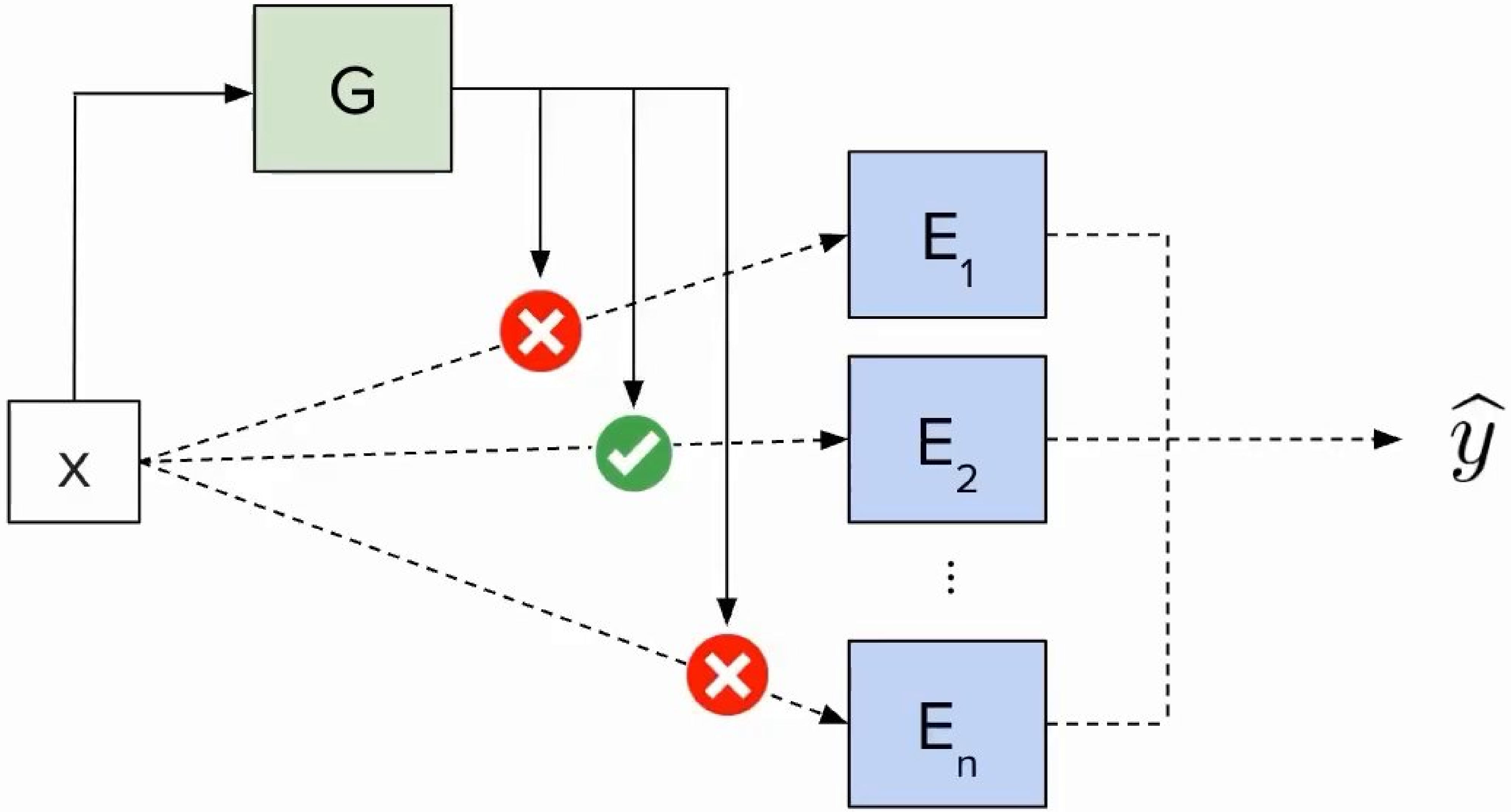
Motivation

Idea. Not all weights are useful in the forward pass




Motivation
x
E1
E2
En
ŷ
Stanford University
Stanford

Motivation



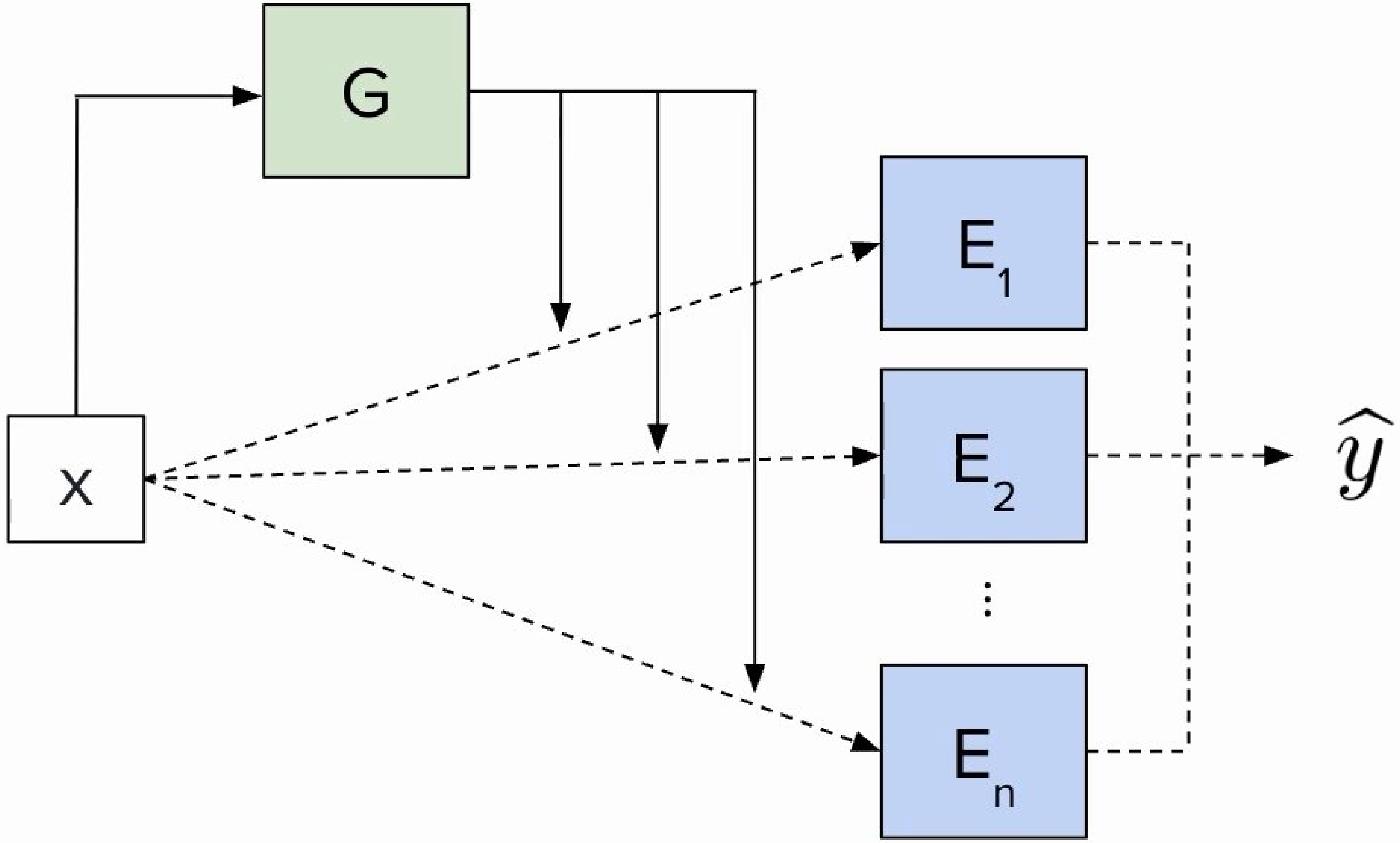
Stanford

ICME


Motivation
x
E2
ŷ
Stanford University
Stanford

Overview of MoEs

MoE = **M**ixture **of** **E**xperts



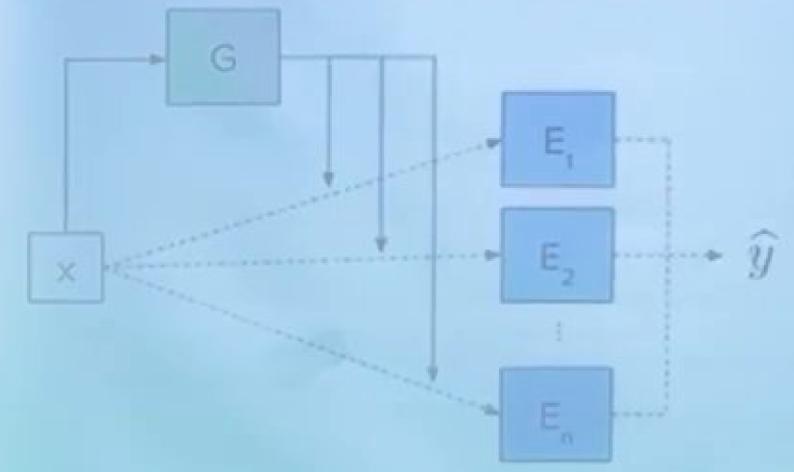
$$\widehat{y} = \sum_{i=1}^{n} G(x)_i E_i(x)$$

Stanford


Overview of MoEs
MoE = Mixture of Experts
G
x
E_1
E_2
E_n
$\widehat{y}$
$\widehat{y} = \sum_{i=1}^{n} G(x)_i E_i(x)$
Stanford University
ICME
Stanford

Overview of MoEs

MoE = **M**ixture **o**f **E**xperts



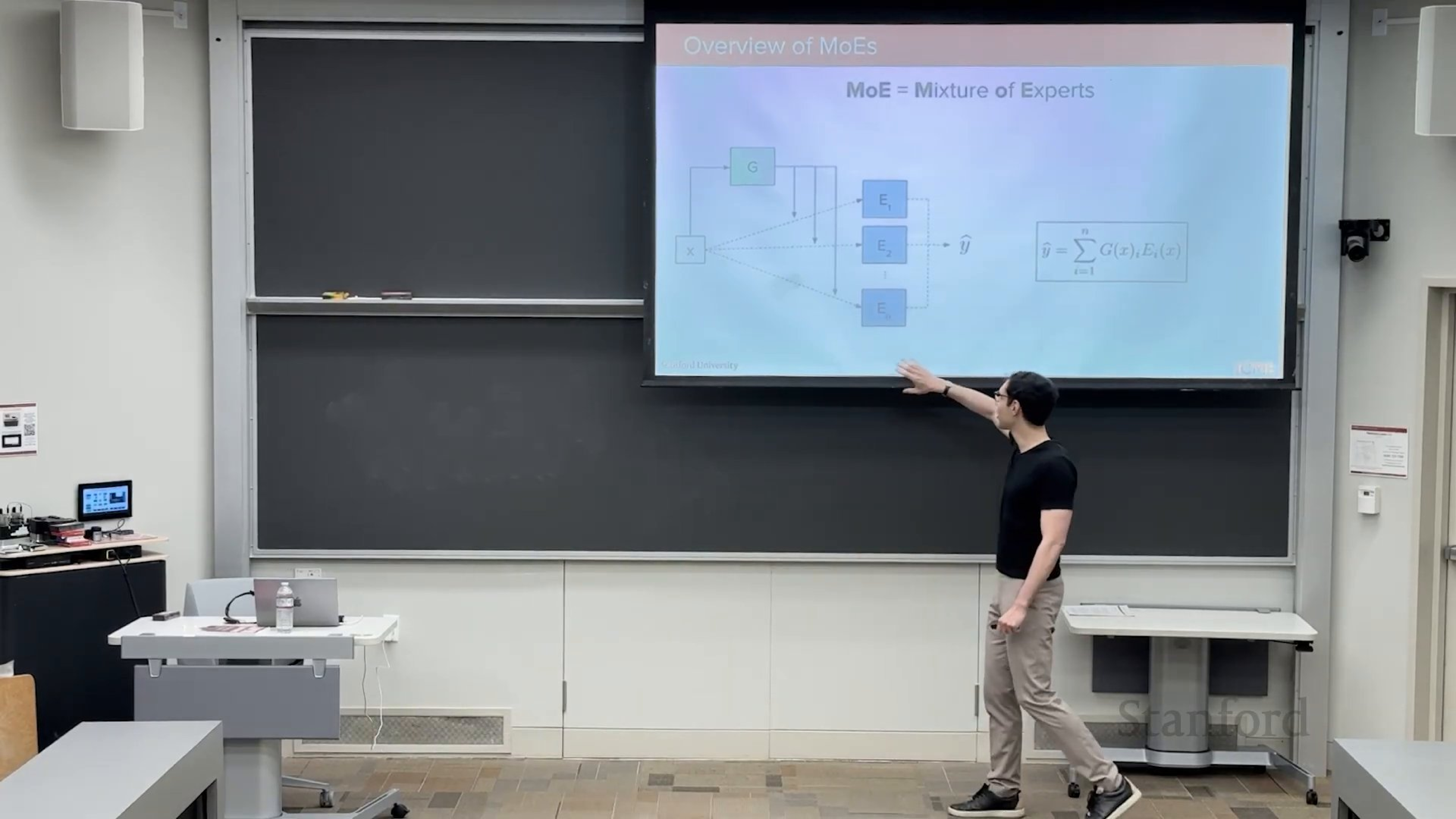
$$\widehat{y} = \sum_{i=1}^{n} G(x)_i E_i(x)$$


Overview of MoEs
MoE = Mixture of Experts
G
x
E1
E2
En
ŷ
$\hat{y} = \sum_{i=1}^{n} G(x)_i E_i(x)$
Stanford University
Stanford


Overview of MoEs
MoE = Mixture of Experts
G
x
E1
E2
En
ŷ
$\hat{y} = \sum_{i=1}^{n} G(x)_i E_i(x)$
Stanford University
Stanford

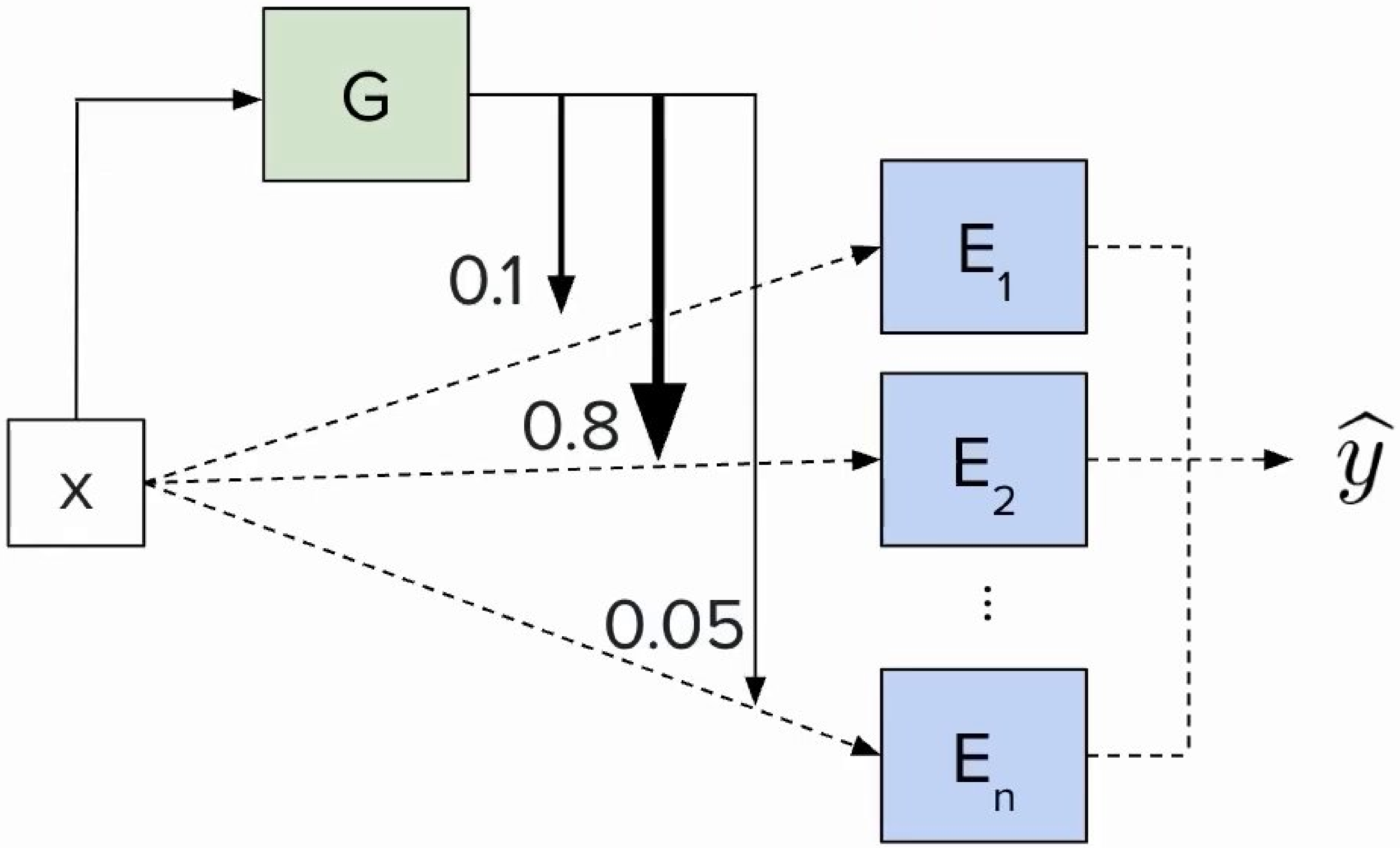

Overview of MoEs
MoE = Mixture of Experts
G
x
E1
E2
En
ŷ
$\hat{y} = \sum_{i=1}^{n} G(x)_i E_i(x)$
Stanford University
Stanford


Overview of MoEs
MoE = Mixture of Experts
G
x
E_1
E_2
E_n
$\widehat{y}$
$\widehat{y} = \sum_{i=1}^{n} G(x)_i E_i(x)$
Stanford University
Stanford


Overview of MoEs
MoE = Mixture of Experts
G
x
E1
E2
En
ŷ
$\hat{y} = \sum_{i=1}^{n} G(x)_i E_i(x)$
Stanford University
Stanford

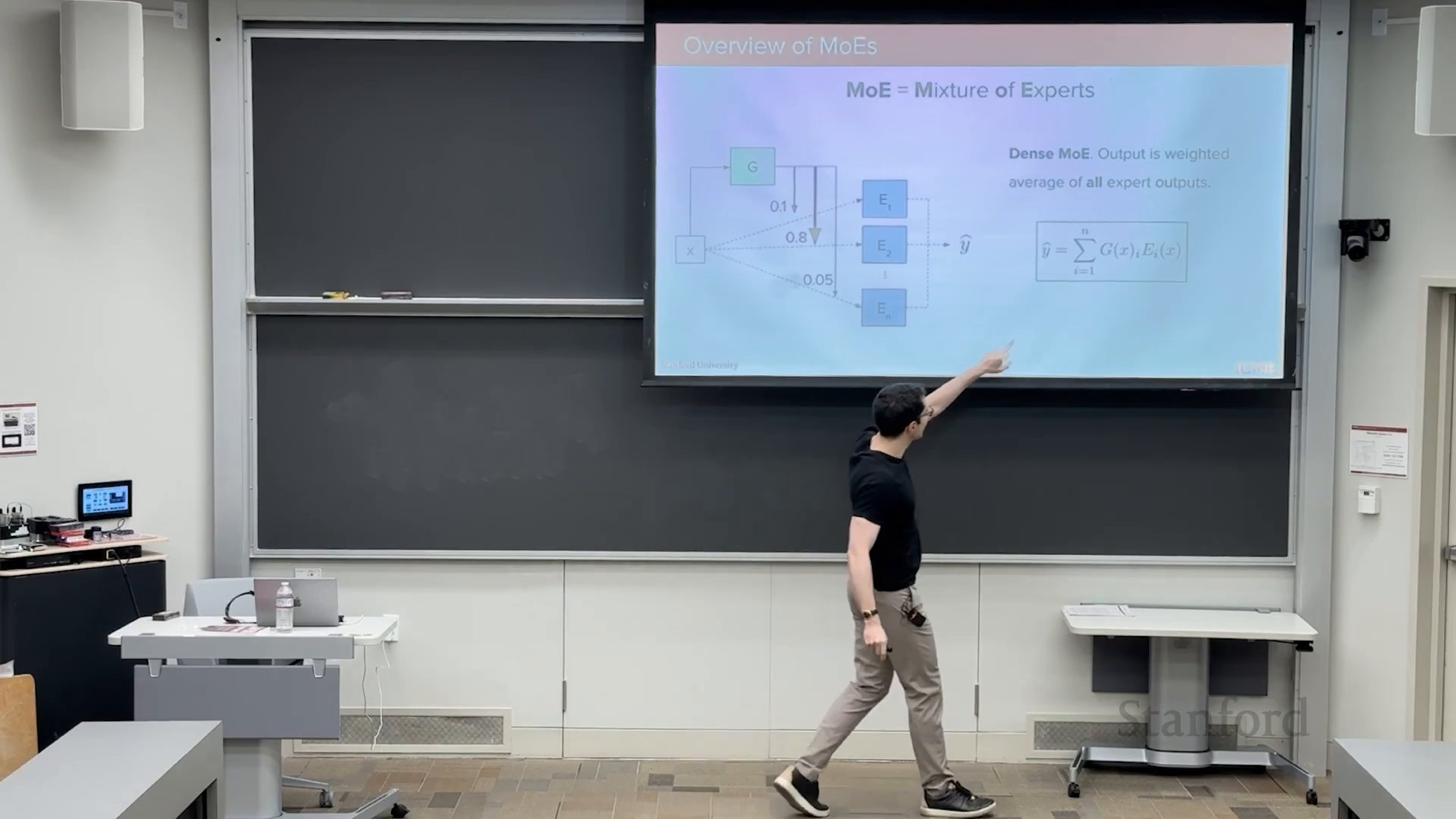
Overview of MoEs

MoE = **M**ixture **of** **E**xperts



Dense MoE. Output is weighted average of **all** expert outputs.

$$\widehat{y} = \sum_{i=1}^{n} G(x)_i E_i(x)$$

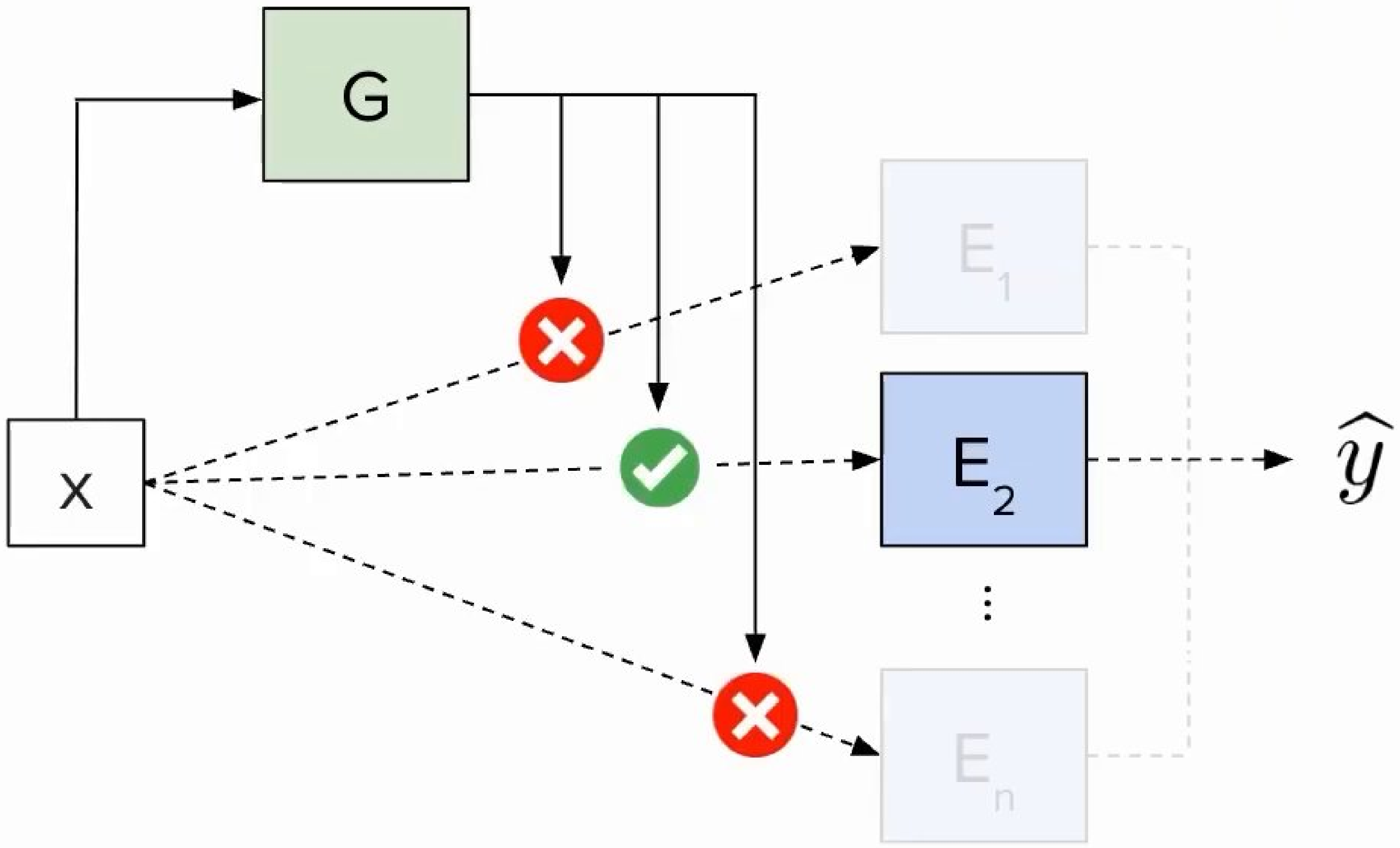


Overview of MoEs
MoE = Mixture of Experts
G
0.1
0.8
0.05
x
E1
E2
En
ŷ
Dense MoE. Output is weighted average of all expert outputs.
$\hat{y} = \sum_{i=1}^{n} G(x)_i E_i(x)$
Stanford University
Stanford


Overview of MoEs
MoE = Mixture of Experts
G
0.1
0.8
0.05
x
E1
E2
En
ŷ
Dense MoE. Output is weighted average of all expert outputs.
$\hat{y} = \sum_{i=1}^{n} G(x)_i E_i(x)$
Stanford University
Stanford

Overview of MoEs

MoE = **M**ixture **of** **E**xperts



Sparse MoE. Output is weighted average of **selected** expert outputs.

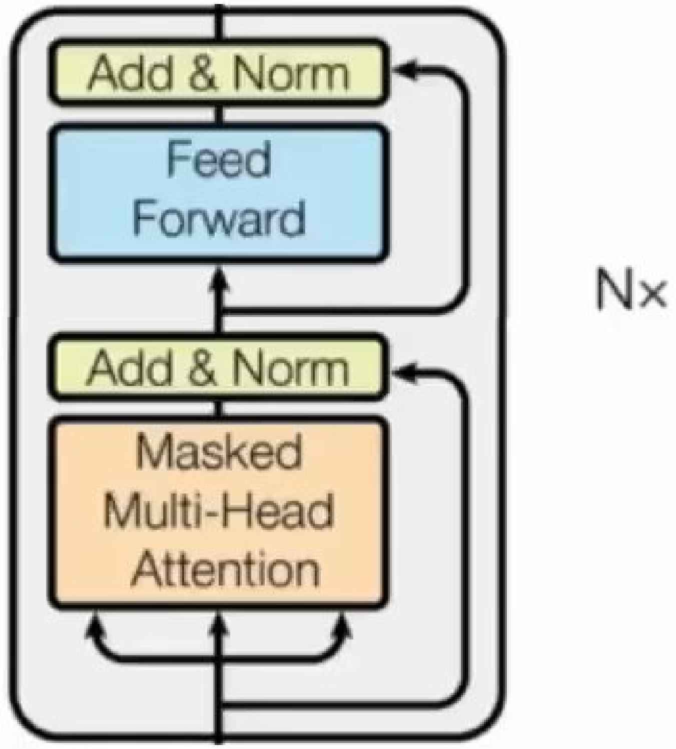
$$\hat{y} = \sum_{i \in \mathcal{I}_k} G(x)_i E_i(x)$$

Via **top-k** selection


Overview of MoEs
MoE = Mixture of Experts
G
x
E2
ŷ
Sparse MoE. Output is weighted average of selected expert outputs.
$\widehat{y} = \sum_{i \in \mathcal{I}_k} G(x)_i E_i(x)$
Via top-k selection
Stanford University
"The Sparsely-Gated Mixture-of-Experts Layer", Shazeer et al., 2017
ICME

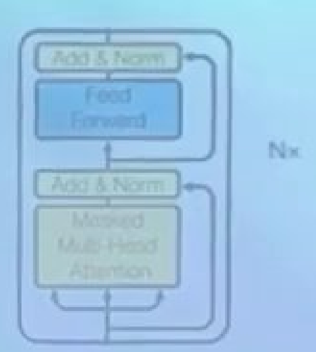

Overview of MoEs
MoE = Mixture of Experts
G
x
E2
ŷ
Sparse MoE. Output is weighted average of selected expert outputs.
$\hat{y} = \sum_{i \in \mathcal{I}_k} G(x)_i E_i(x)$
Via top-k selection
Stanford

MoE in Transformer-based models

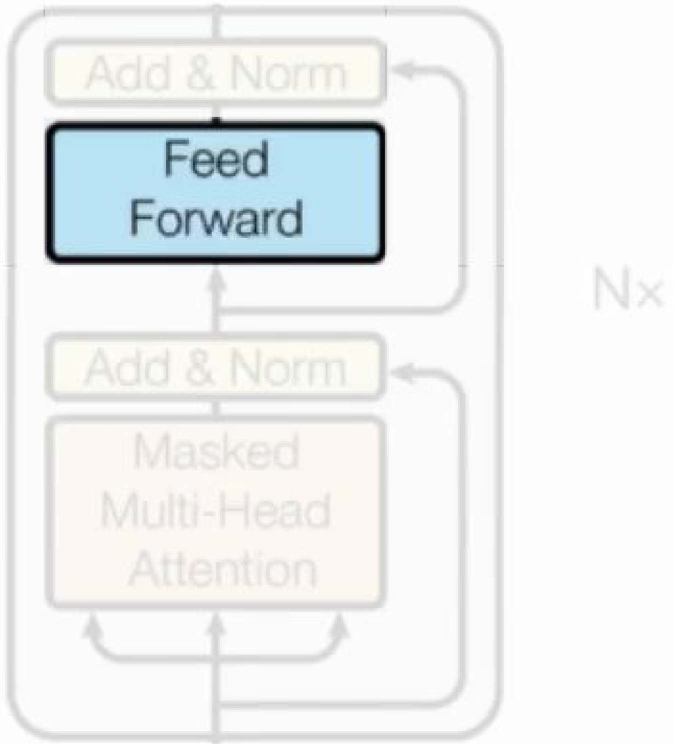


Stanford

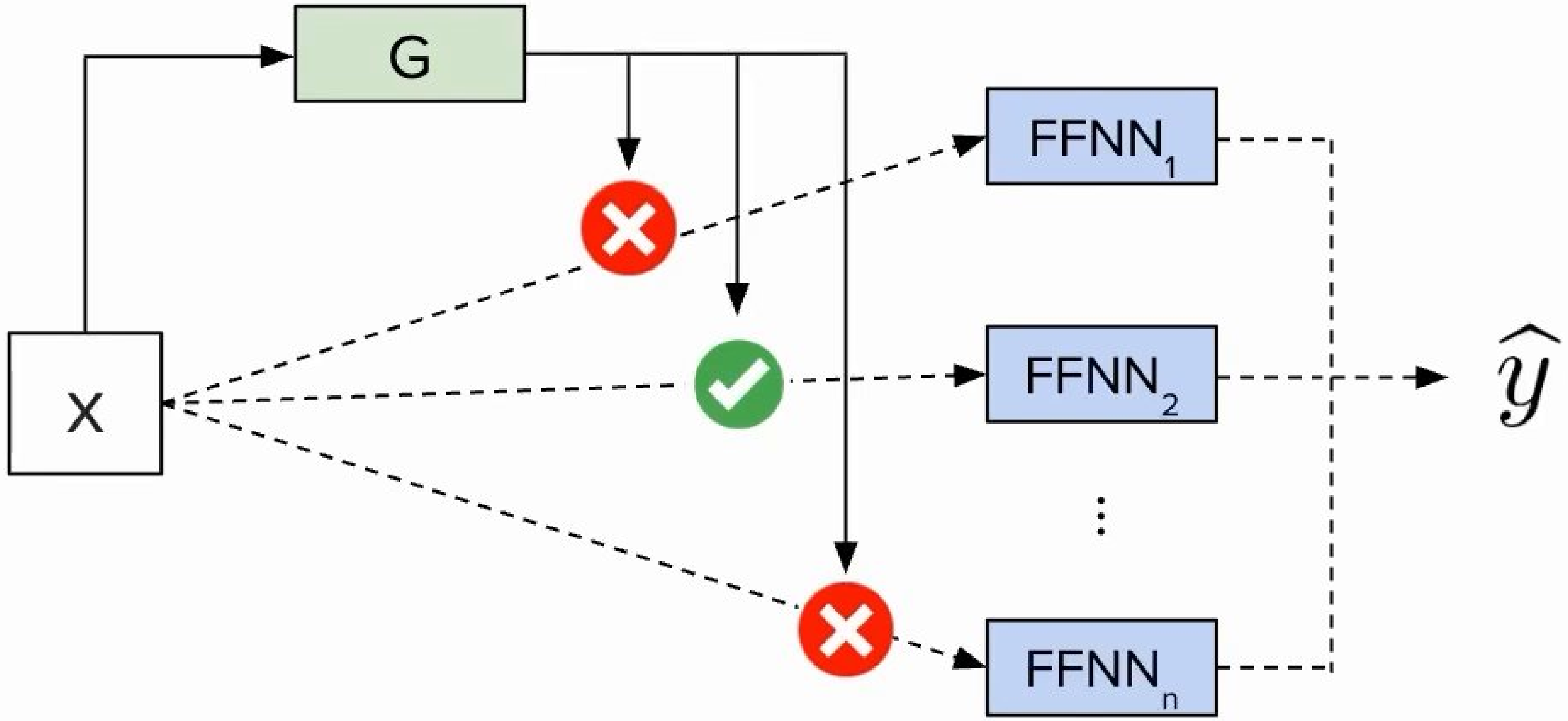
MoE in Transformer-based models



MoE in Transformer-based models

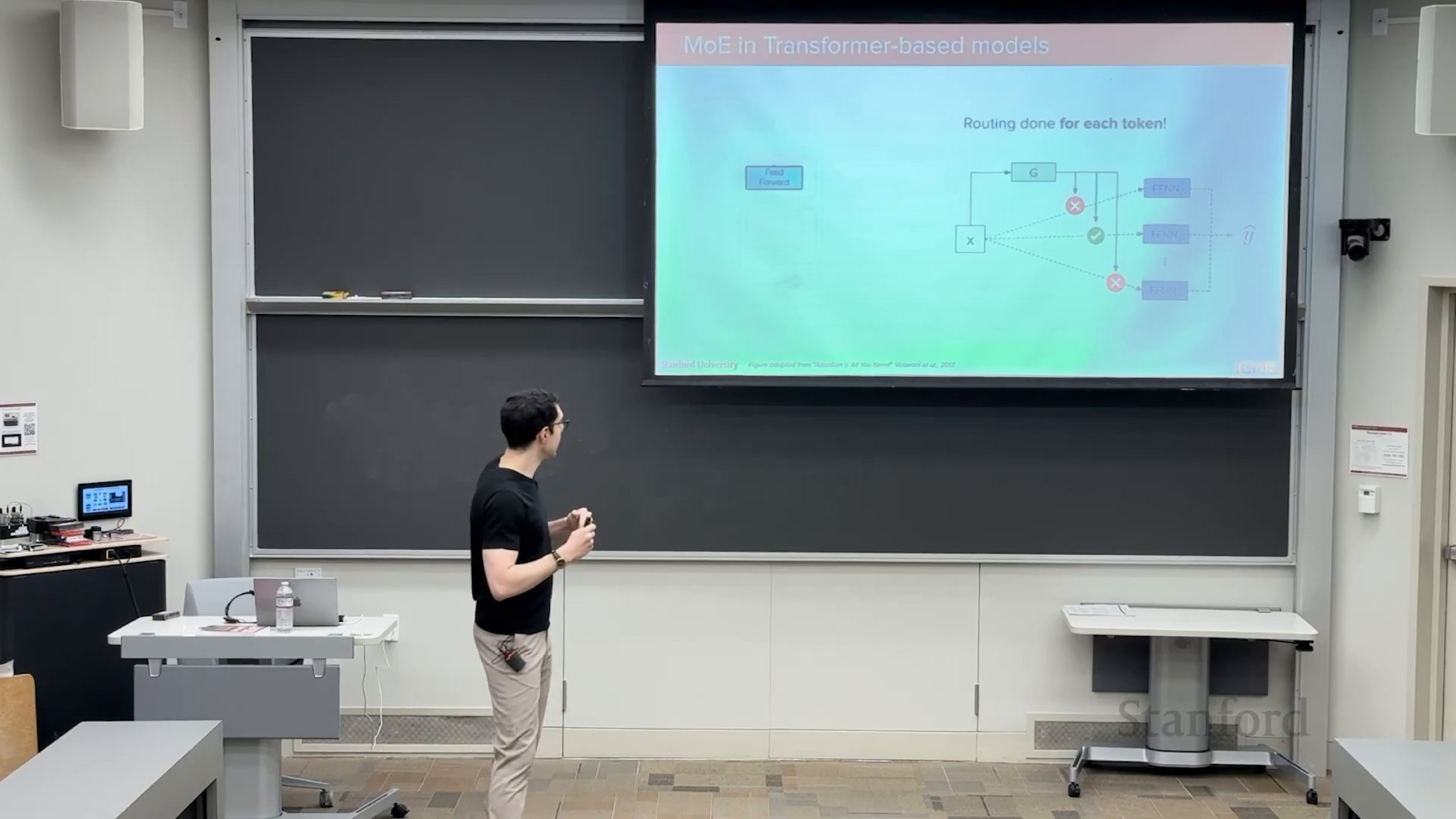


Routing done **for each token**!



Stanford

Figure adapted from "Attention is All You Need", Vaswani et al., 2017.


MoE in Transformer-based models
Routing done for each token!
Feed Forward
G
x
FFNN
FFNN
FFNN
$\hat{y}$
Stanford University
Figure adapted from "Attention is All You Need", Vaswani et al., 2017
ICME
Stanford

Training challenges include routing collap

Symptom. Same expert gets selected most of the time.

"routing collapse"

Stanford

 "Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity", Fedus et al., 2021.


Training challenges include routing collapse
Symptom. Same expert gets selected most of the time.
"routing collapse"
Stanford University
"Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity", Fedus et al., 2021
Stanford

Training challenges include routing collap

Symptom. Same expert gets selected most of the time.

"routing collapse"

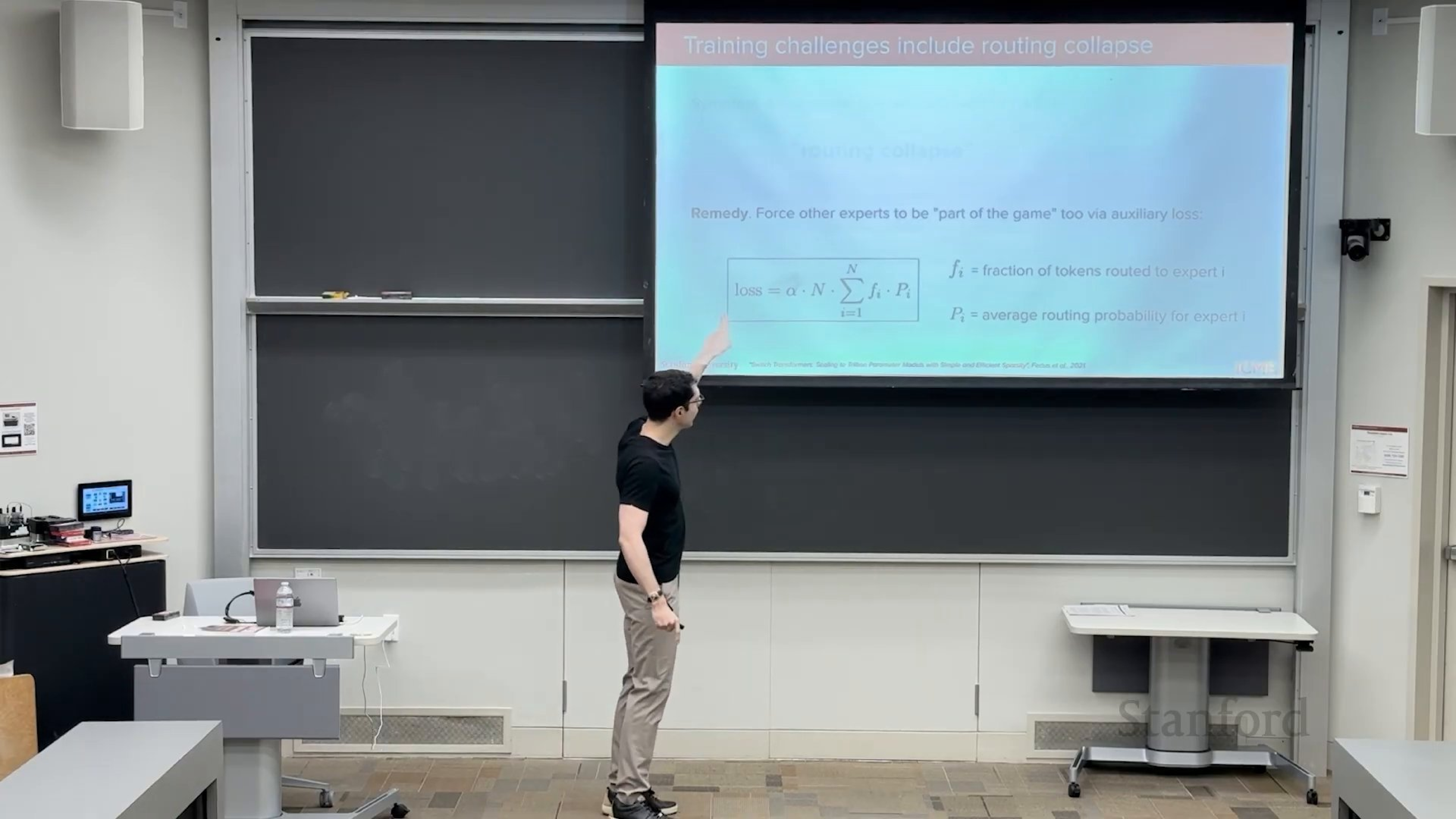
Remedy. Force other experts to be "part of the game" too via auxiliary loss:

$$\boxed{\text{loss} = \alpha \cdot N \cdot \sum_{i=1}^{N} f_i \cdot P_i}$$

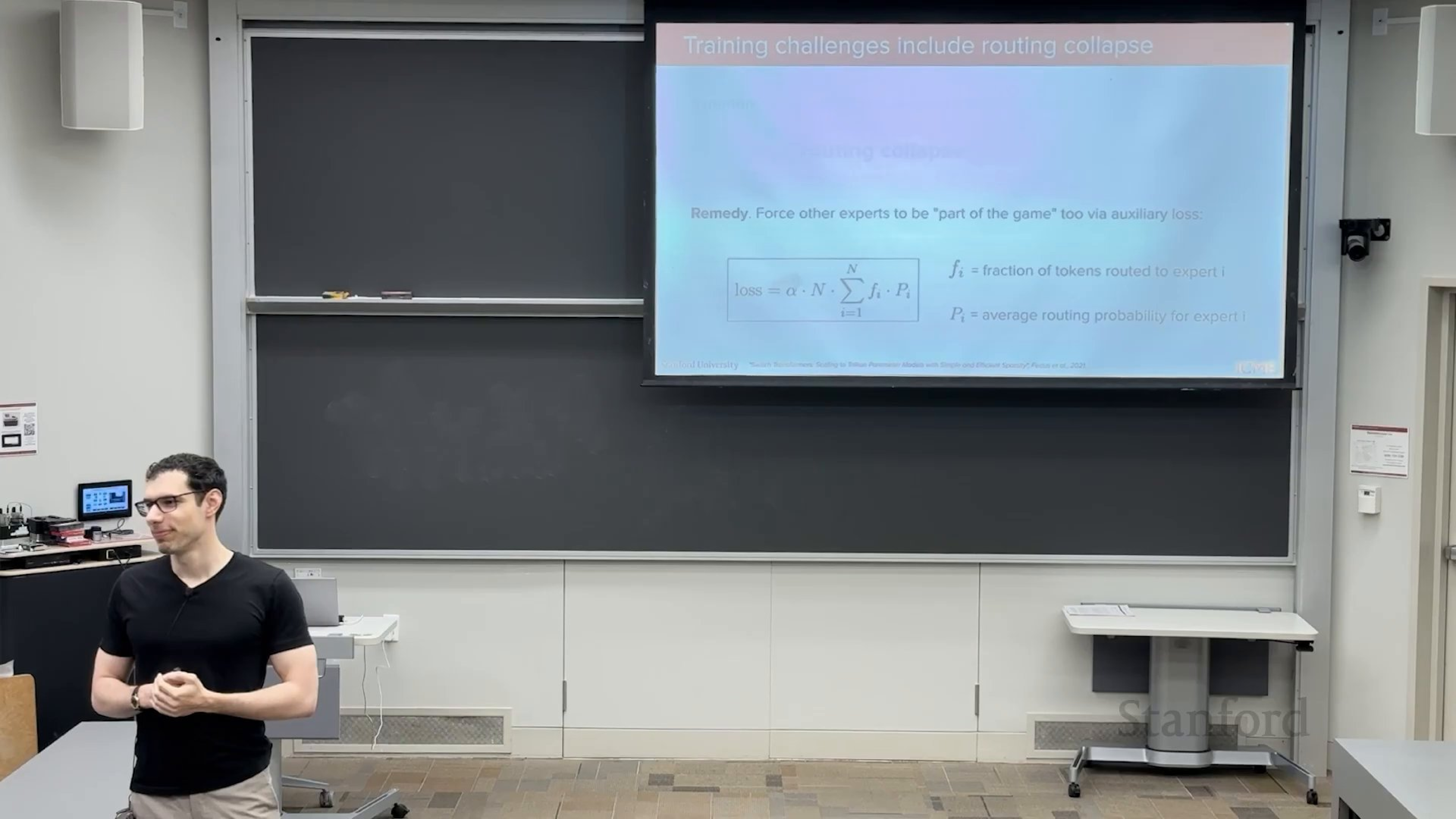
f_i = fraction of tokens routed to expert i

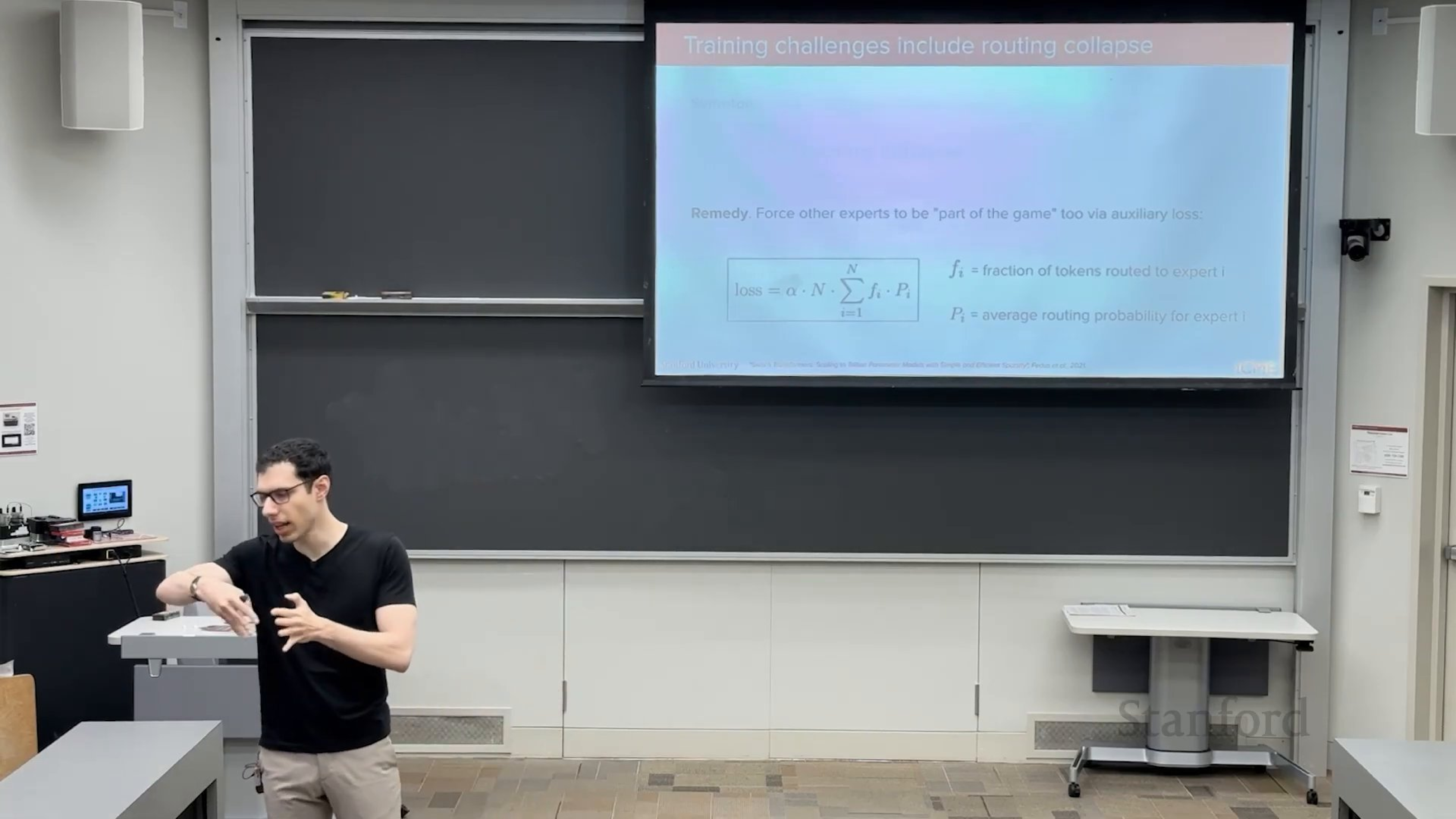
P_i = average routing probability for expert i

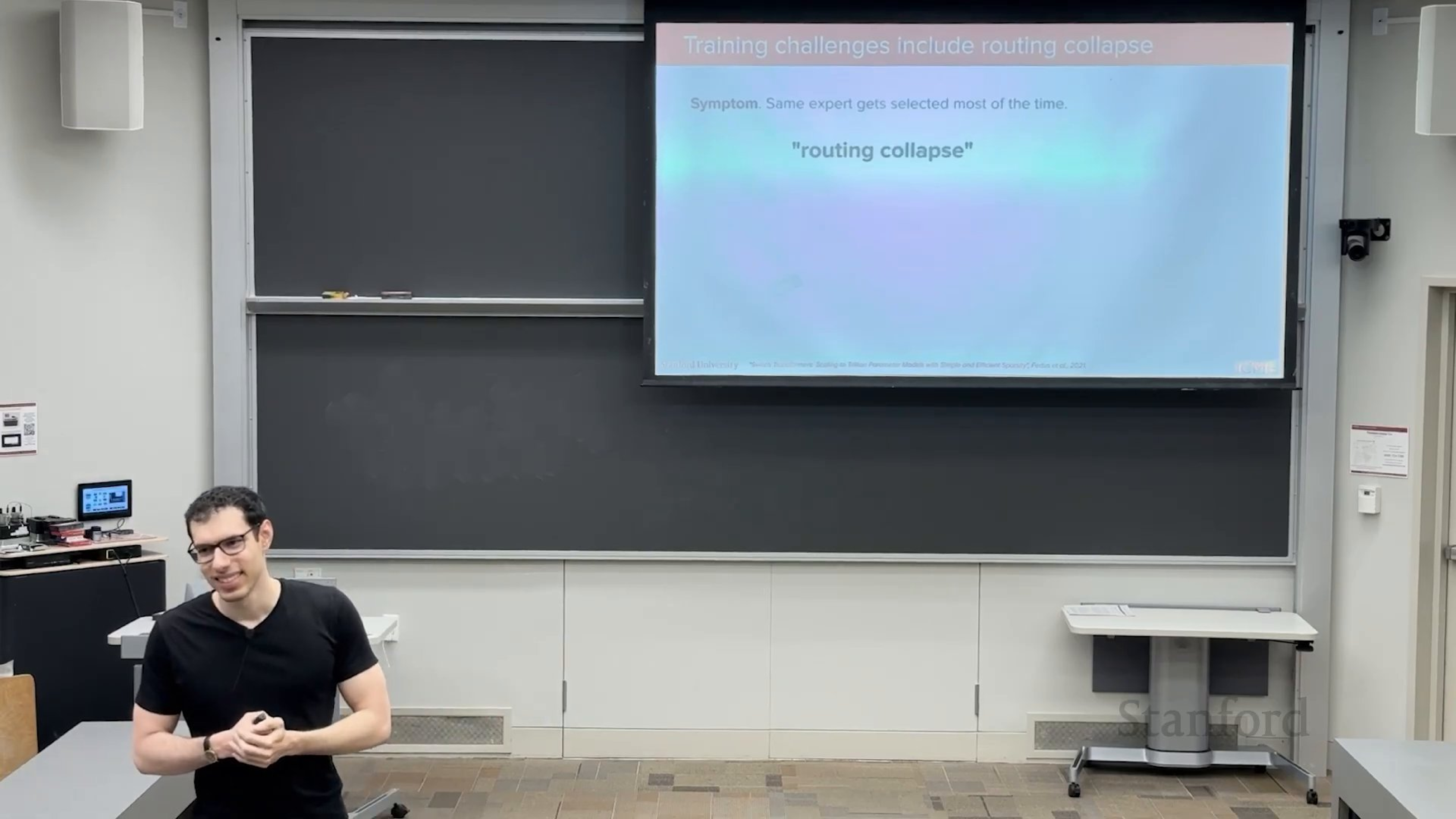
Stanford

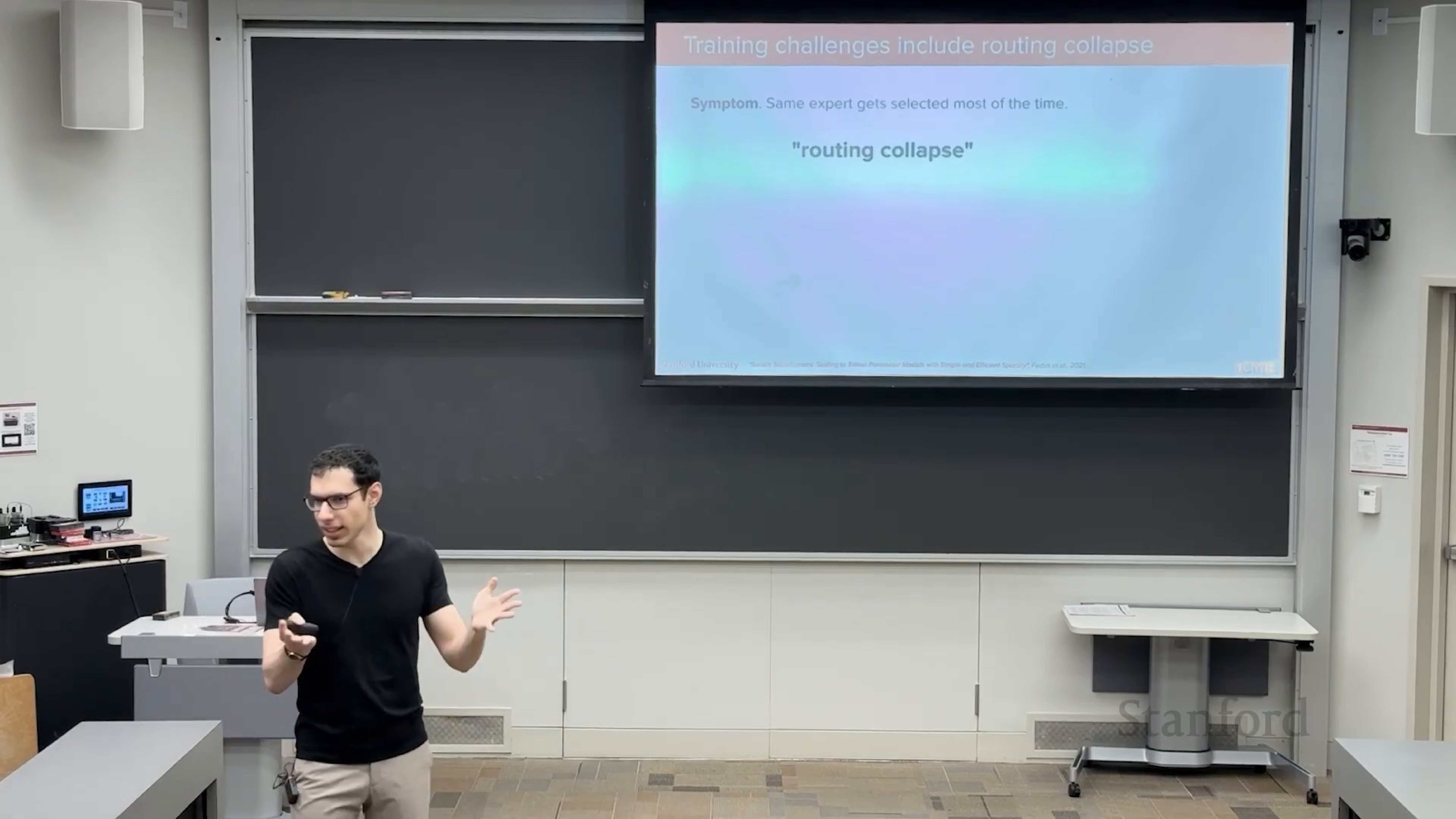


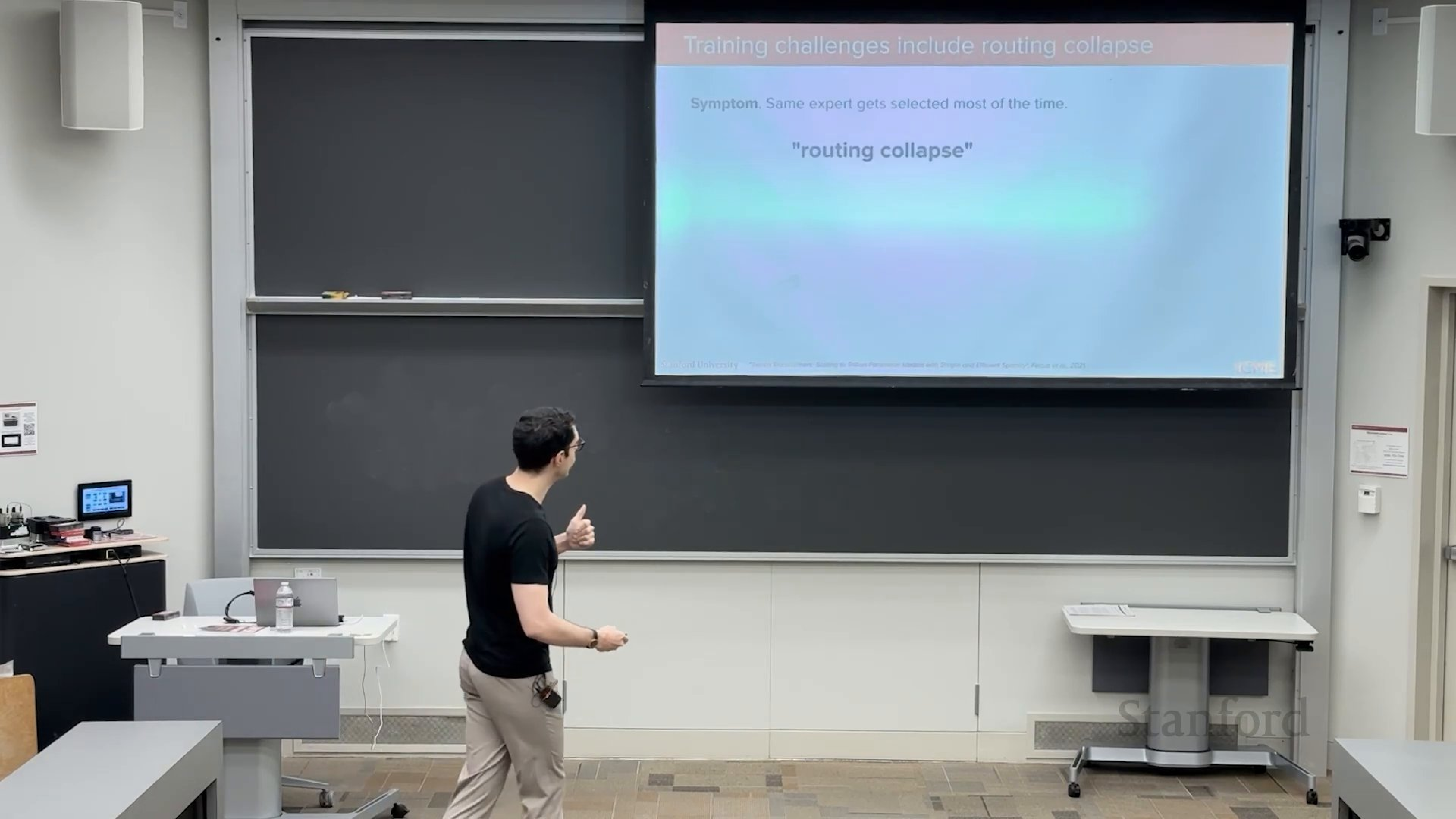
Training challenges include routing collapse
Remedy. Force other experts to be "part of the game" too via auxiliary loss:
$\text{loss} = \alpha \cdot N \cdot \sum_{i=1}^{N} f_i \cdot P_i$
f_i = fraction of tokens routed to expert i
P_i = average routing probability for expert i
Stanford


Training challenges include routing collapse
Remedy. Force other experts to be "part of the game" too via auxiliary loss:
$\boxed{\text{loss} = \alpha \cdot N \cdot \sum_{i=1}^{N} f_i \cdot P_i}$
f_i = fraction of tokens routed to expert i
P_i = average routing probability for expert i
Stanford University
"Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity", Fedus et al., 2021
ICME
Stanford

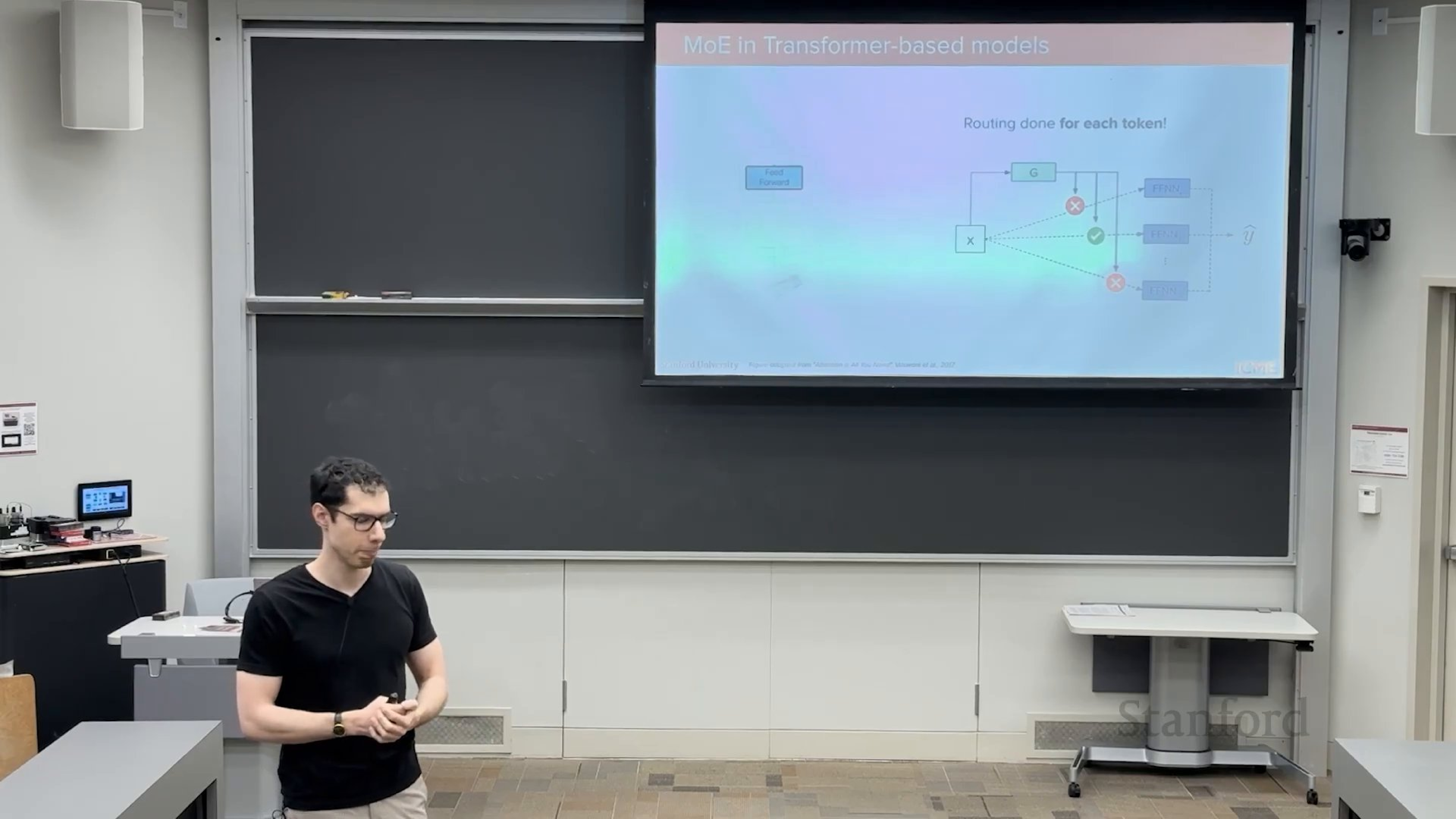

Training challenges include routing collapse
Remedy. Force other experts to be "part of the game" too via auxiliary loss:
$\boxed{\text{loss} = \alpha \cdot N \cdot \sum_{i=1}^{N} f_i \cdot P_i}$
f_i = fraction of tokens routed to expert i
P_i = average routing probability for expert i
Stanford University
"Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity", Fedus et al., 2021
ICME
Stanford

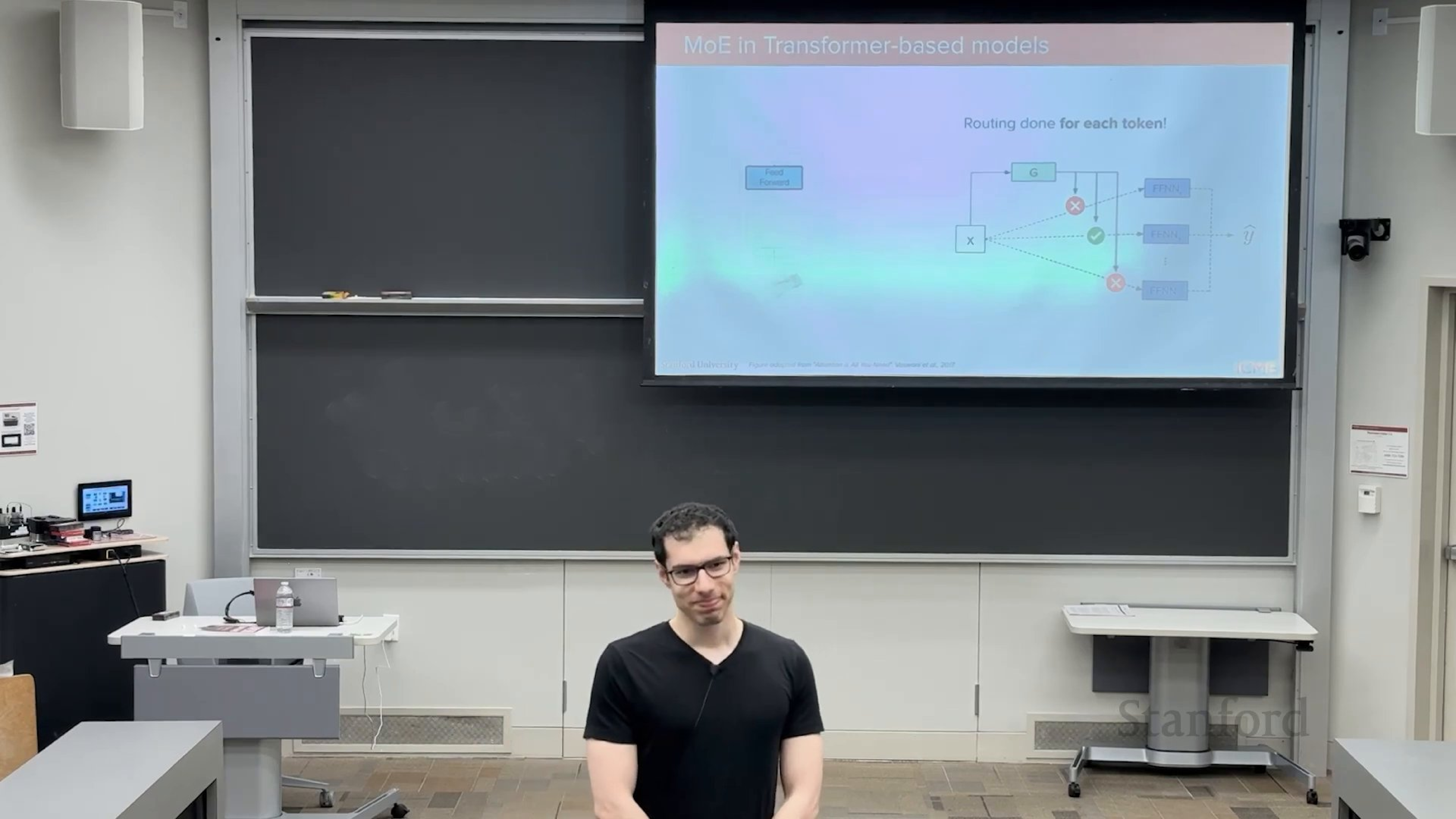

Training challenges include routing collapse
Symptom. Same expert gets selected most of the time.
"routing collapse"
Stanford University
Scaling to Trillion Parameter Models with Simple and Efficient Sparsity", Fedus et al., 2021
Stanford

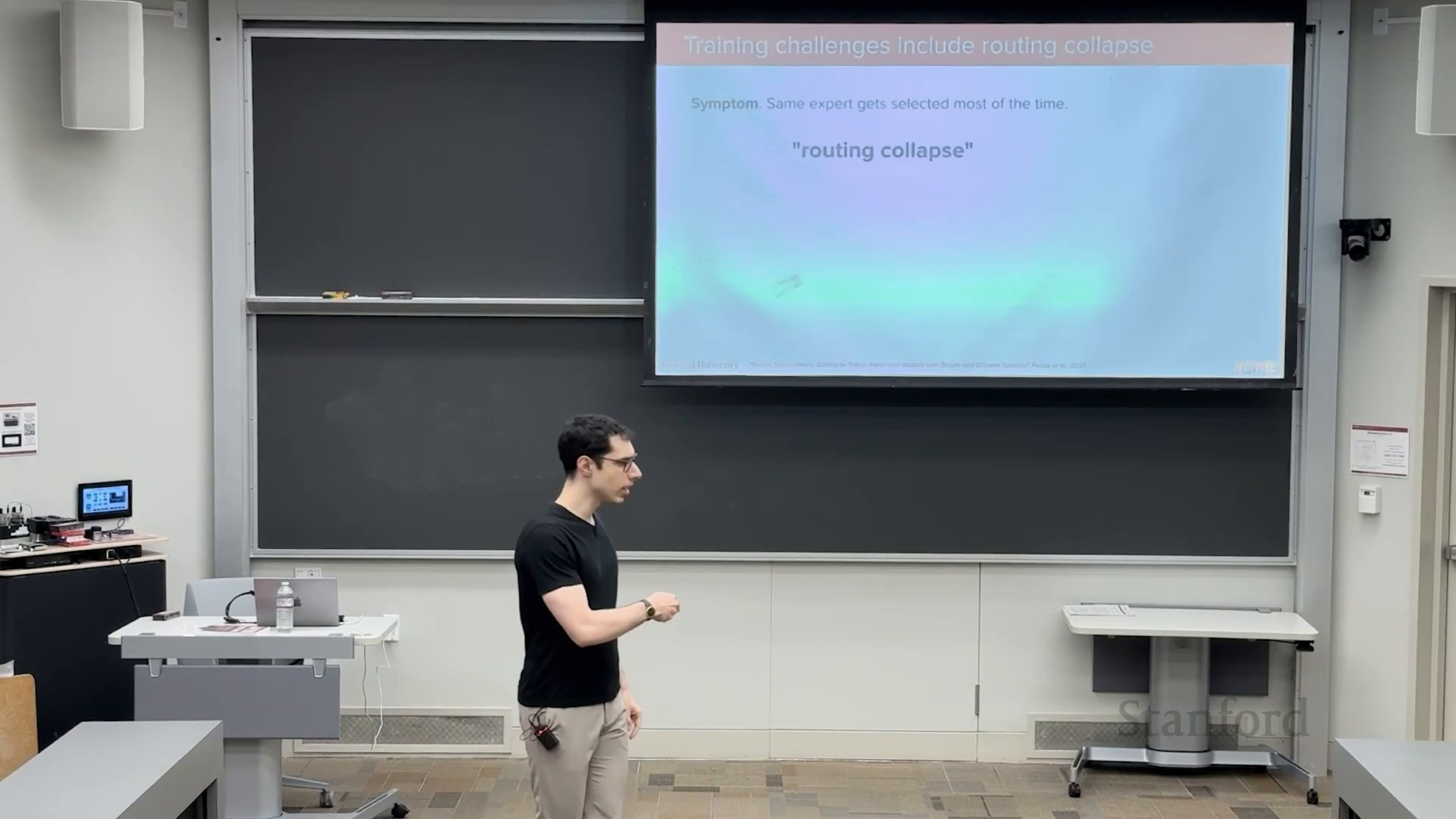

Training challenges include routing collapse
Symptom. Same expert gets selected most of the time.
"routing collapse"
Stanford

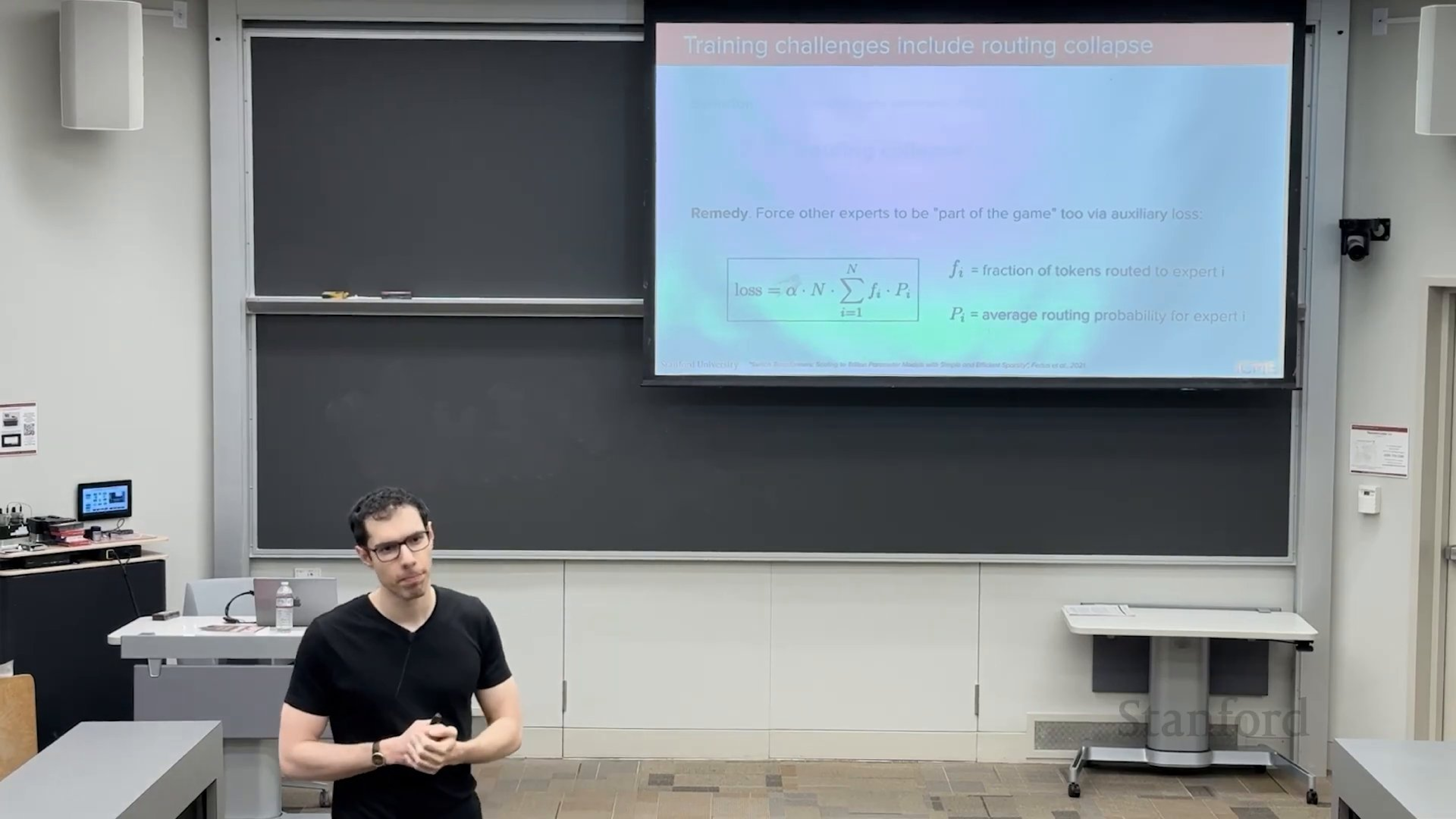

Training challenges include routing collapse
Symptom. Same expert gets selected most of the time.
"routing collapse"
Stanford University
ICME
Stanford

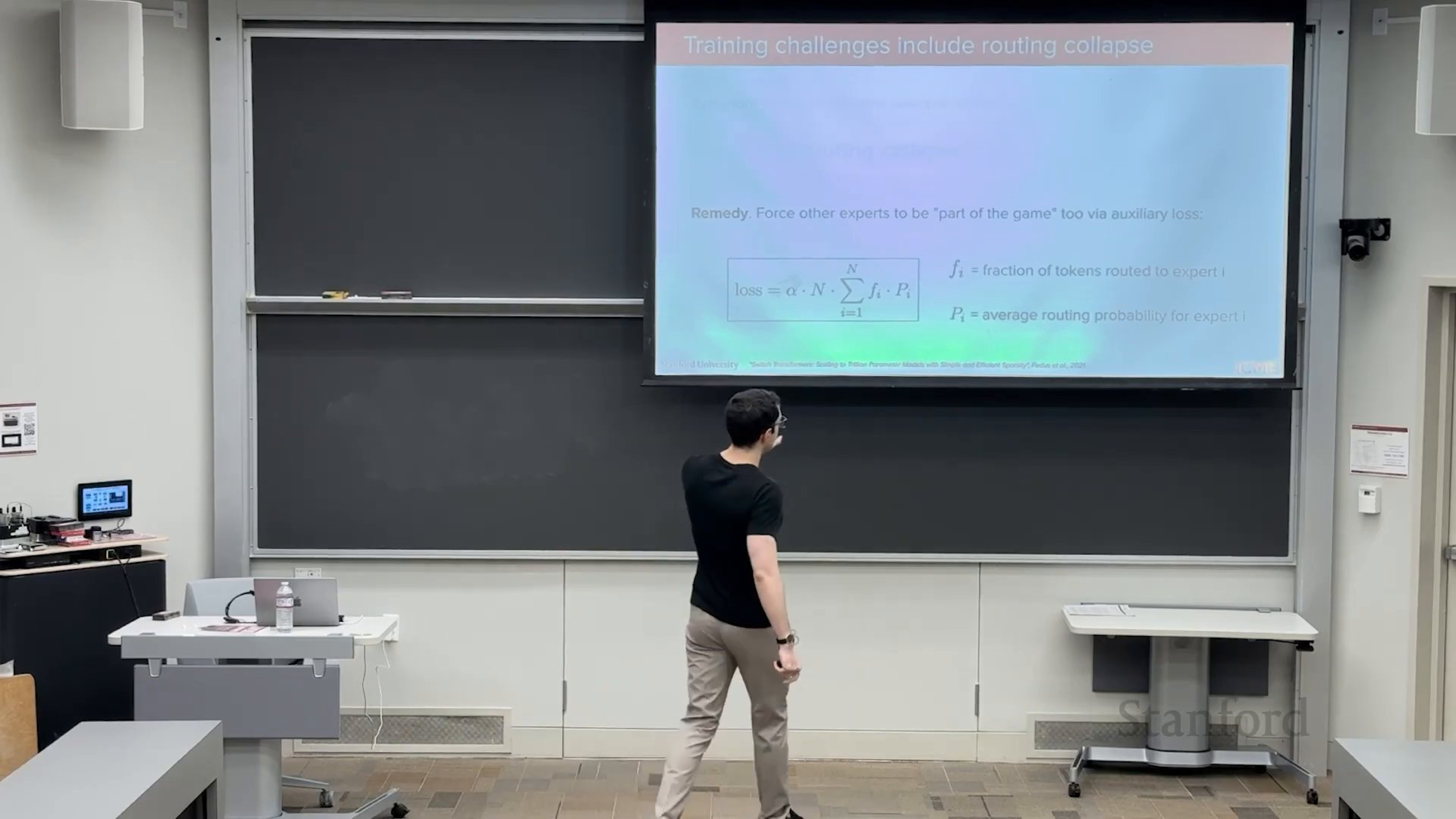

MoE in Transformer-based models
Routing done **for each token!**
Feed Forward
G
X
FFNN
FFNN
FFNN
Stanford University
ICME
Stanford

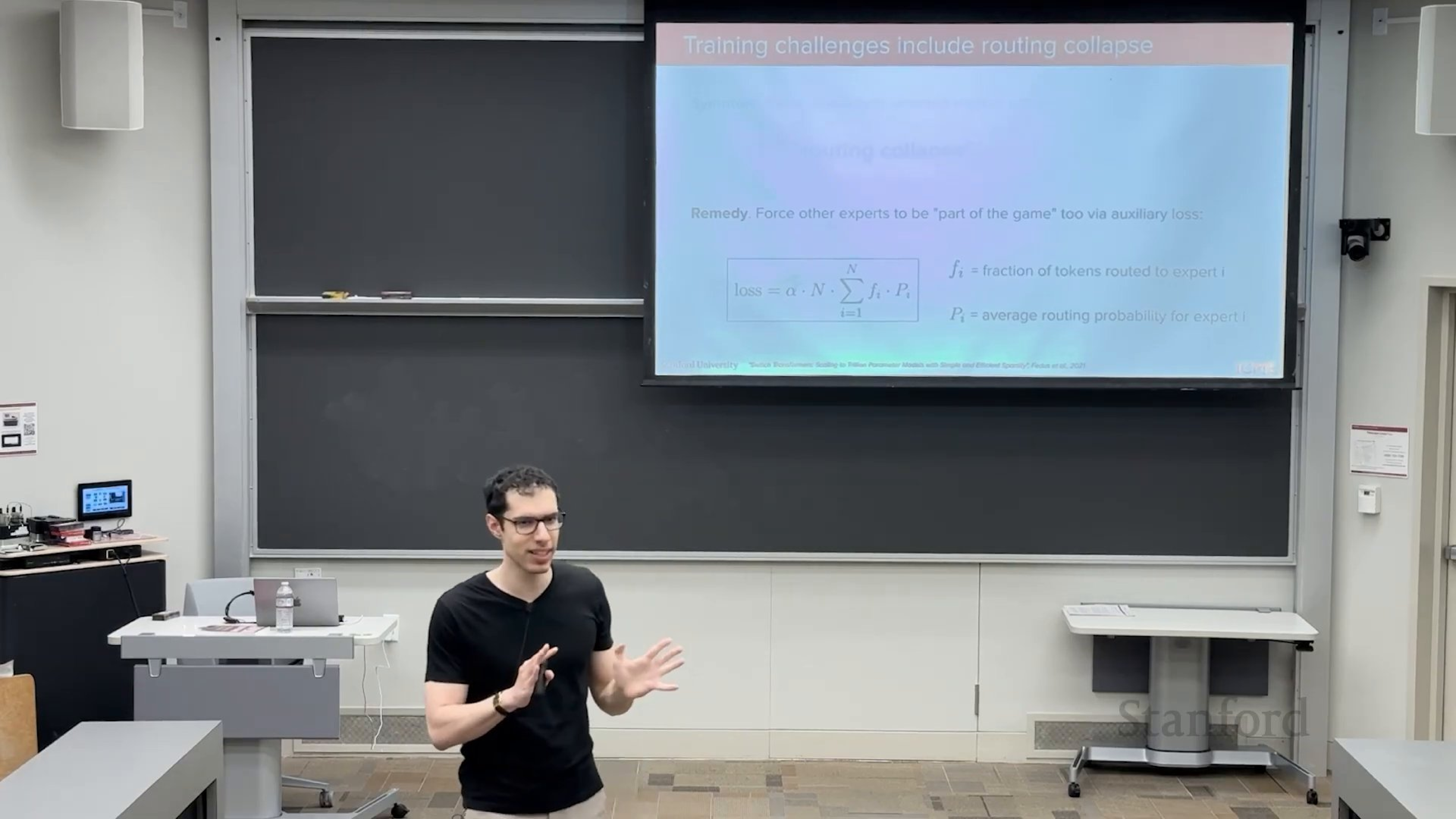

MoE in Transformer-based models
Routing done **for each token!**
Feed Forward
G
X
FFNN
FFNN
FFNN
$\hat{y}$
Stanford University
Stanford

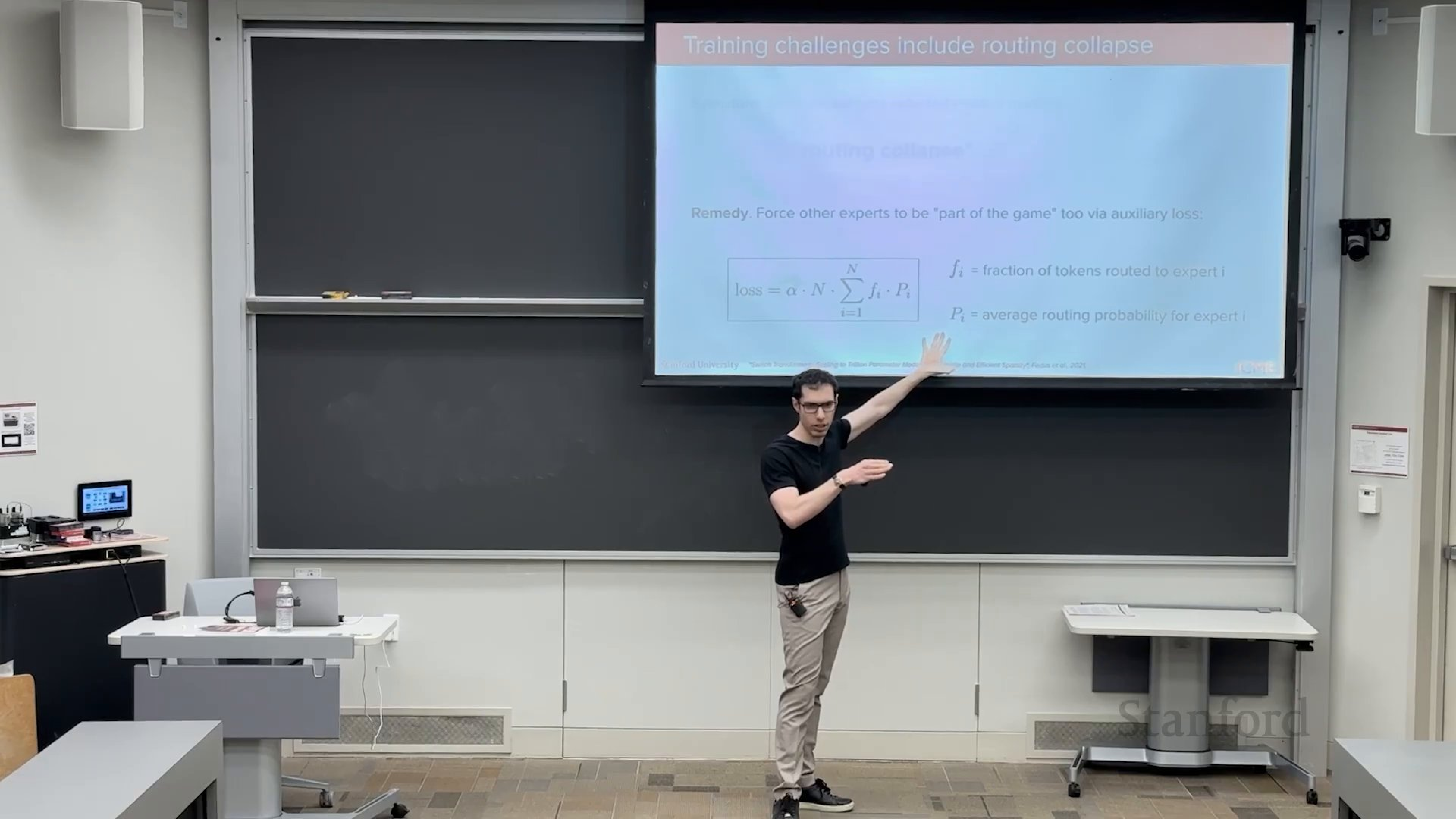

MoE in Transformer-based models
Routing done **for each token!**
Feed Forward
G
X
FFNN
FFNN
FFNN
$\hat{y}$
Stanford University
Figure adapted from "Attention is All You Need", Vaswani et al., 2017
ICME
Stanford

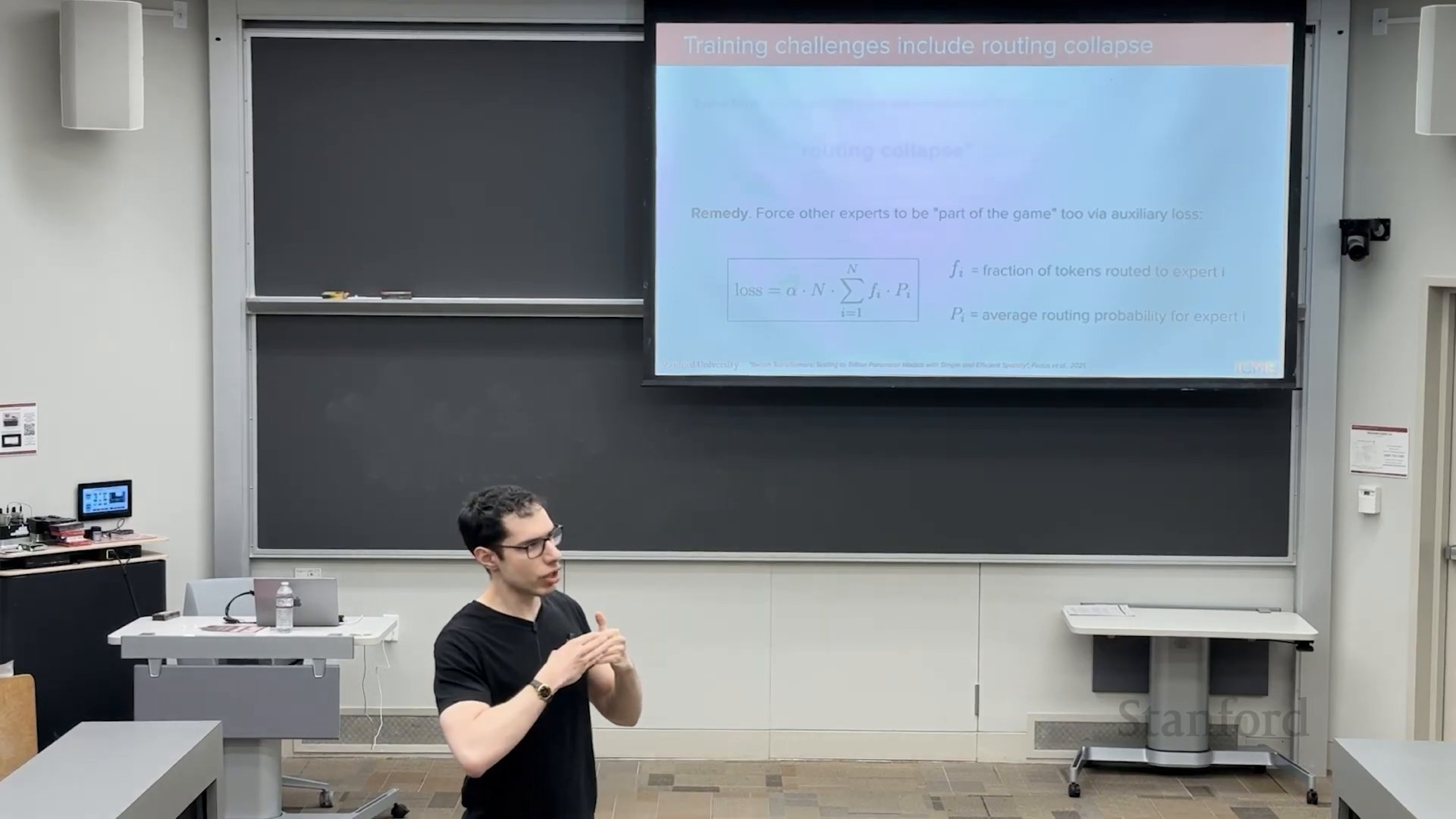

Training challenges include routing collapse
Symptom. Same expert gets selected most of the time.
"routing collapse"
Stanford University
ICME
Stanford

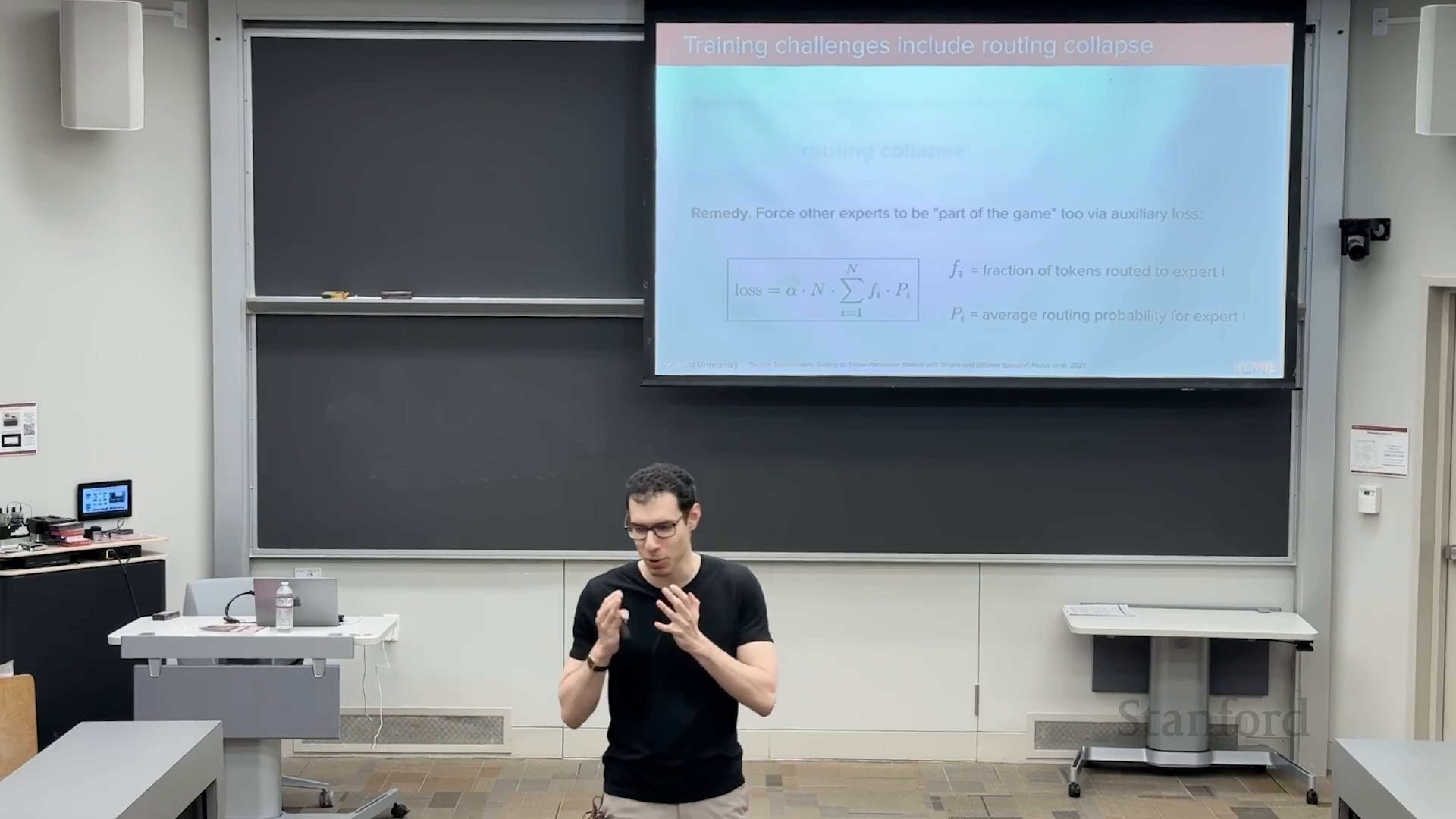

Training challenges include routing collapse
Remedy. Force other experts to be "part of the game" too via auxiliary loss:
$$\boxed{\text{loss} = \alpha \cdot N \cdot \sum_{i=1}^{N} f_i \cdot P_i}$$
f_i = fraction of tokens routed to expert i
P_i = average routing probability for expert i
Stanford University
Stanford

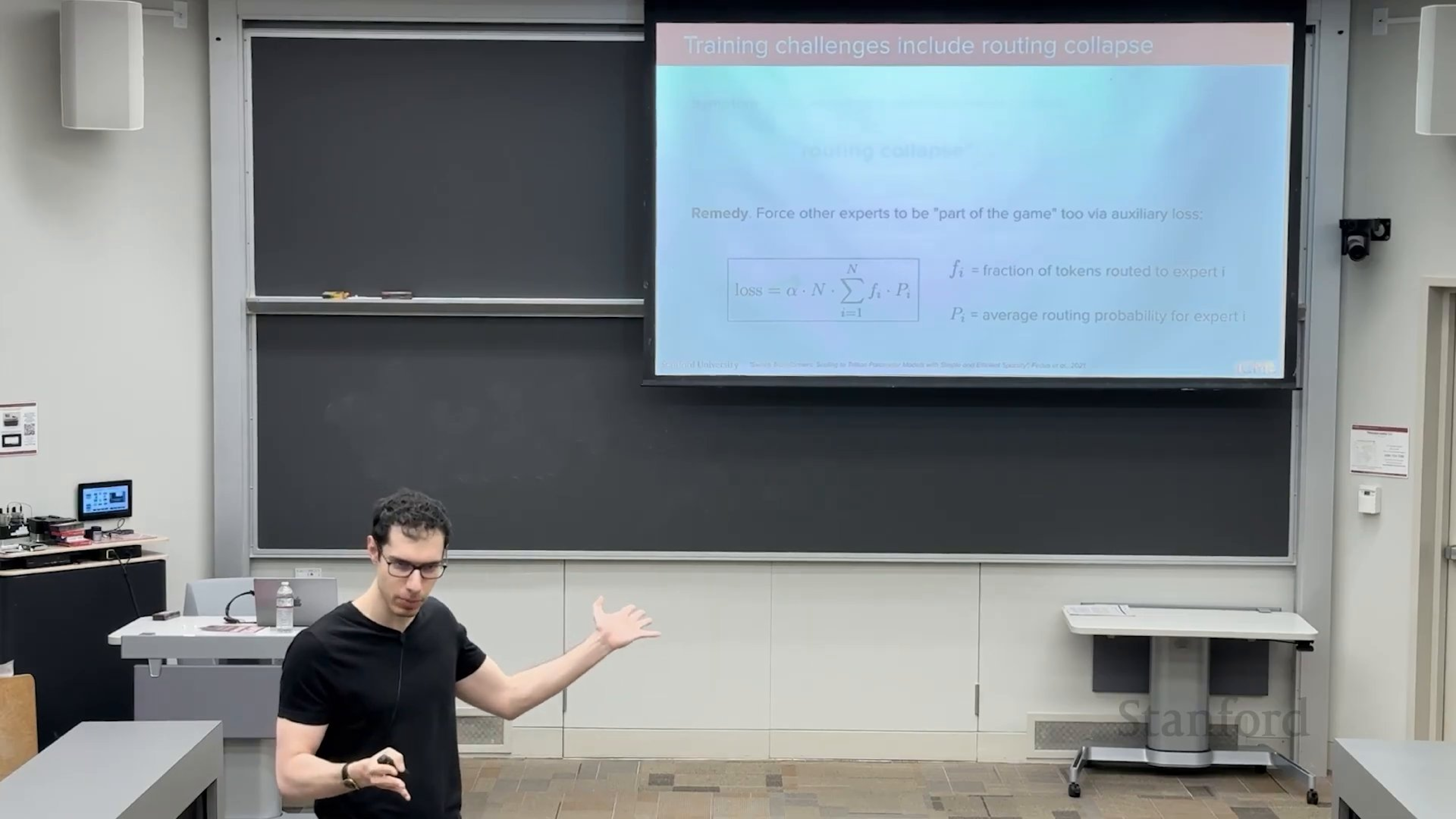

Training challenges include routing collapse
Remedy. Force other experts to be "part of the game" too via auxiliary loss:
$\text{loss} = \alpha \cdot N \cdot \sum_{i=1}^{N} f_i \cdot P_i$
f_i = fraction of tokens routed to expert i
P_i = average routing probability for expert i
Stanford University
"Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity", Fedus et al., 2021
ICME
Stanford

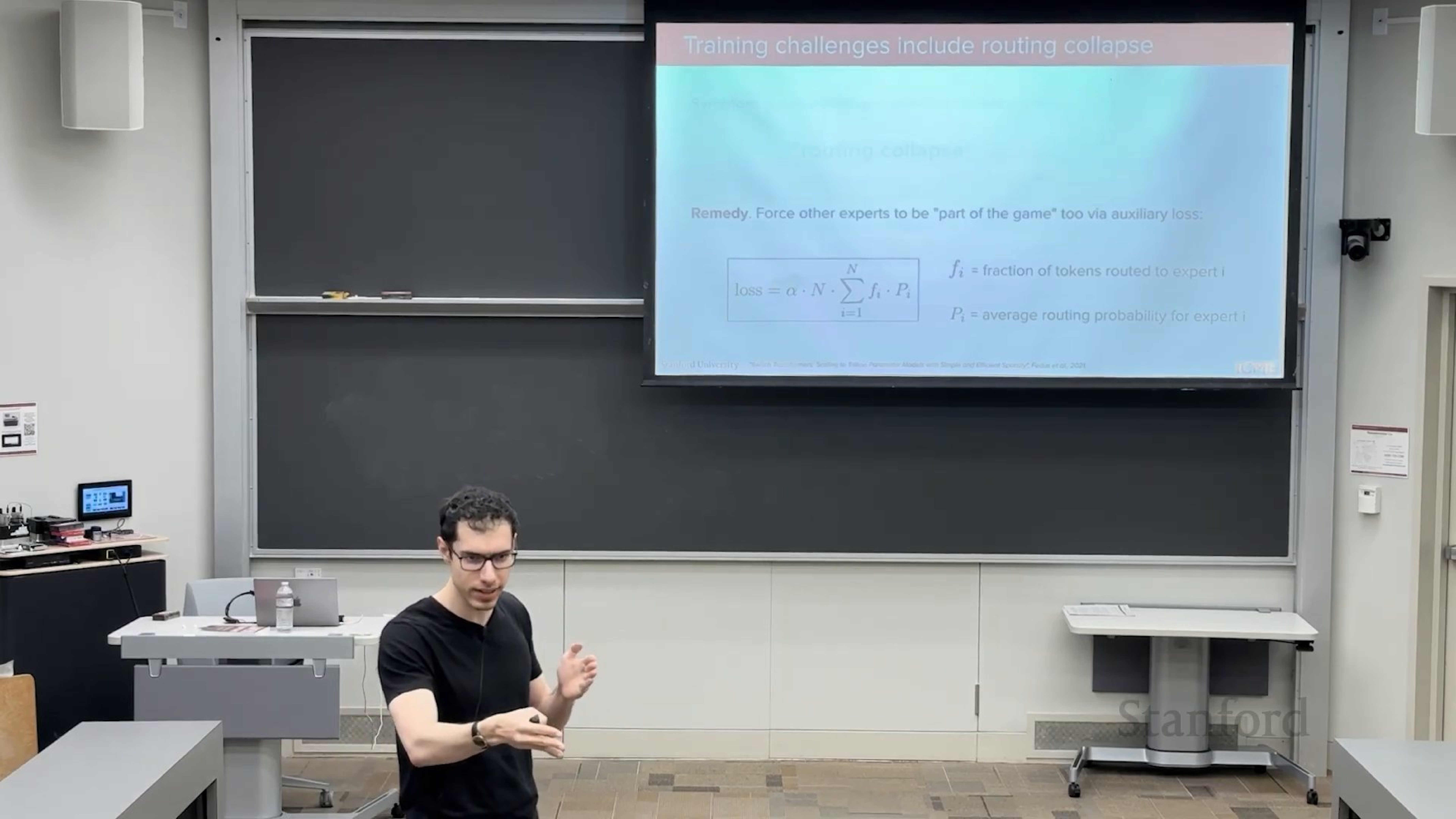

Training challenges include routing collapse
Remedy. Force other experts to be "part of the game" too via auxiliary loss:
$\text{loss} = \alpha \cdot N \cdot \sum_{i=1}^{N} f_i \cdot P_i$
f_i = fraction of tokens routed to expert i
P_i = average routing probability for expert i
Stanford

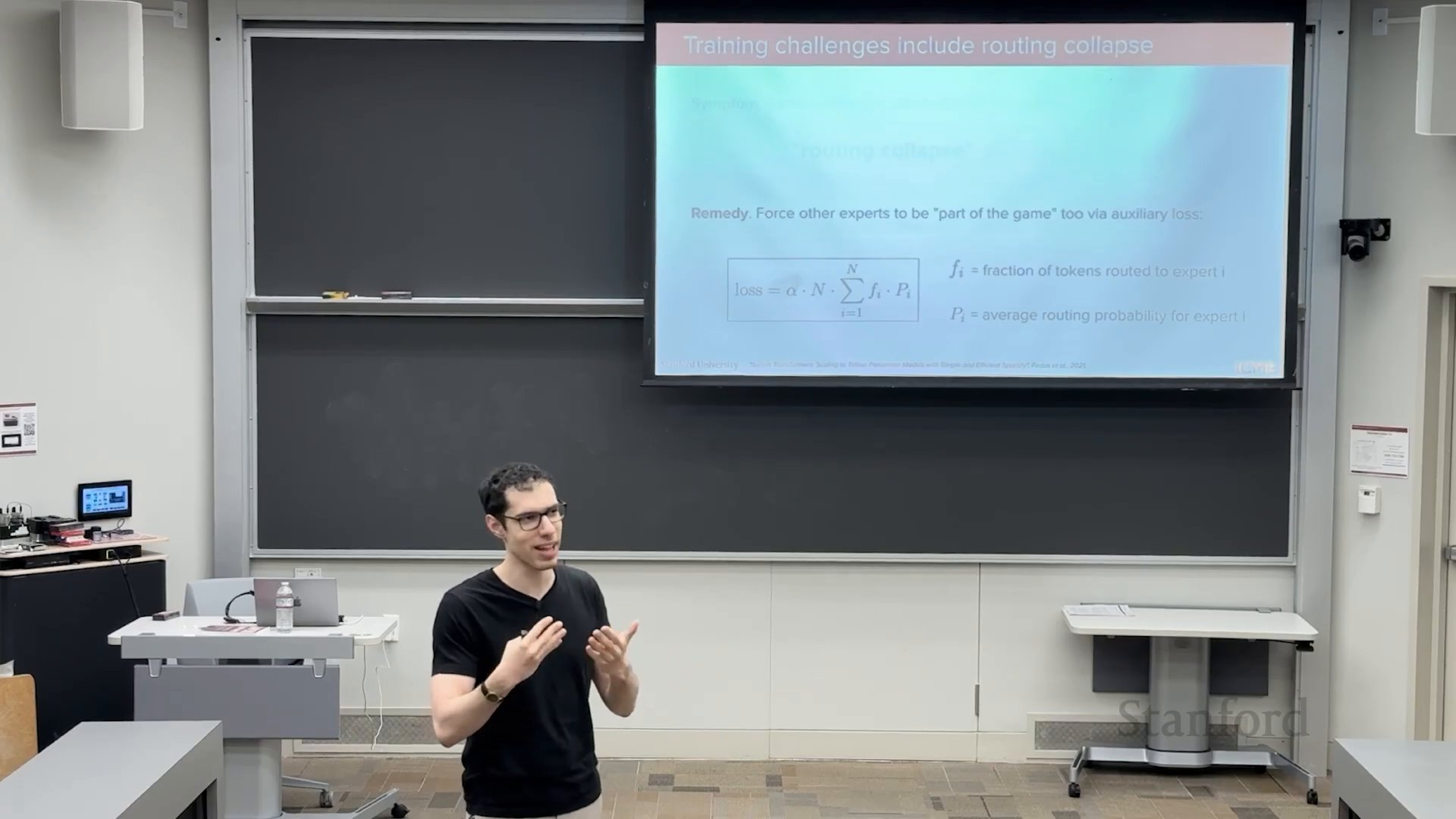

Training challenges include routing collapse
Remedy. Force other experts to be "part of the game" too via auxiliary loss:
$\text{loss} = \alpha \cdot N \cdot \sum_{i=1}^{N} f_i \cdot P_i$
f_i = fraction of tokens routed to expert i
P_i = average routing probability for expert i
Stanford University
ICME
Stanford


Training challenges include routing collapse
"routing collapse"
Remedy. Force other experts to be "part of the game" too via auxiliary loss:
$$\boxed{\text{loss} = \alpha \cdot N \cdot \sum_{i=1}^{N} f_i \cdot P_i}$$
f_i = fraction of tokens routed to expert i
P_i = average routing probability for expert i
Stanford University
"Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity", Fedus et al., 2021
ICME
Stanford

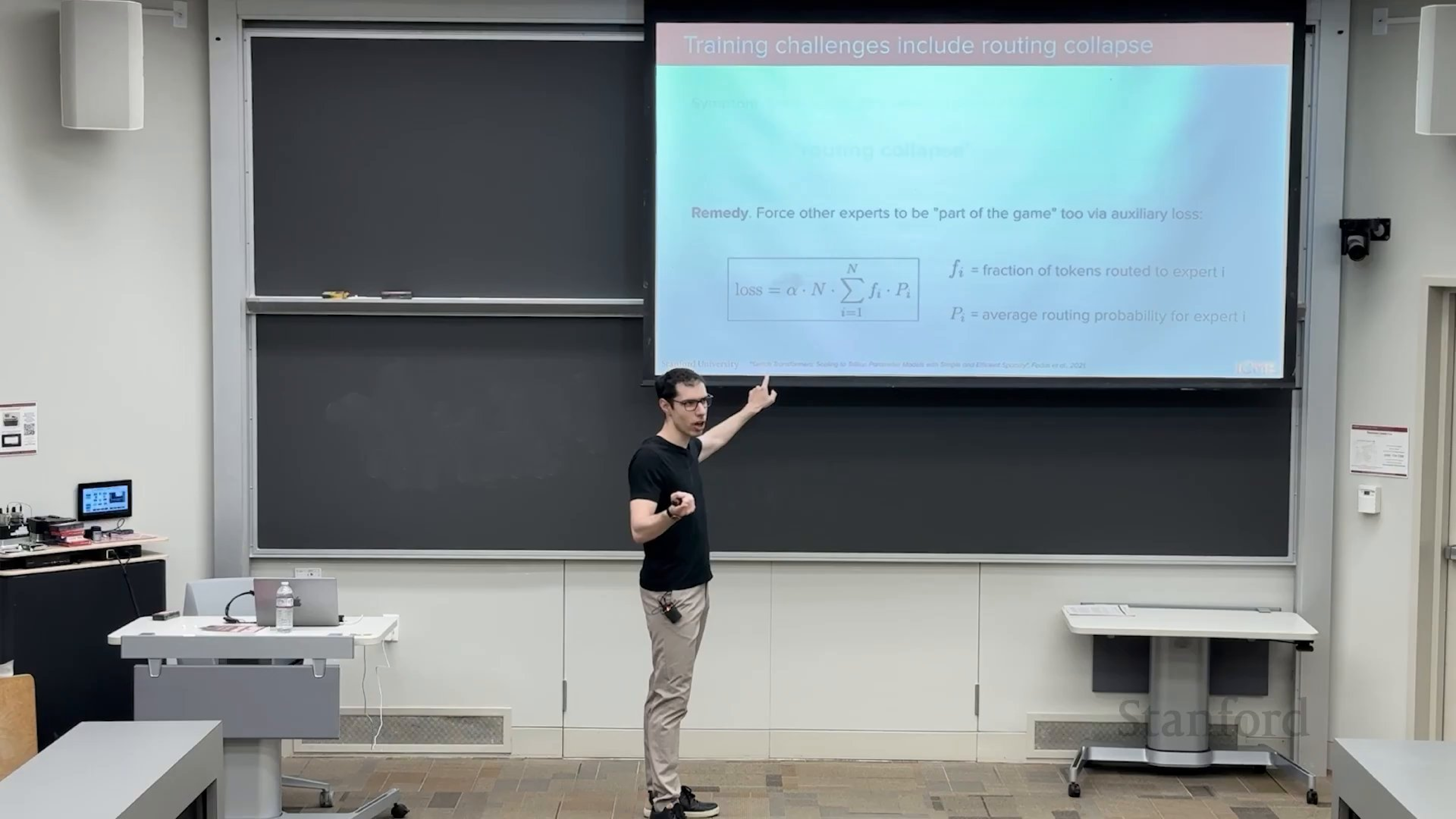

Training challenges include routing collapse
"routing collapse"
Remedy. Force other experts to be "part of the game" too via auxiliary loss:
$\text{loss} = \alpha \cdot N \cdot \sum_{i=1}^{N} f_i \cdot P_i$
f_i = fraction of tokens routed to expert i
P_i = average routing probability for expert i
Stanford University
"Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity", Fedus et al., 2021
Stanford


Training challenges include routing collapse
routing collapse
Remedy. Force other experts to be "part of the game" too via auxiliary loss:
$\text{loss} = \alpha \cdot N \cdot \sum_{i=1}^{N} f_i \cdot P_i$
f_i = fraction of tokens routed to expert i
P_i = average routing probability for expert i
Stanford University
Stanford

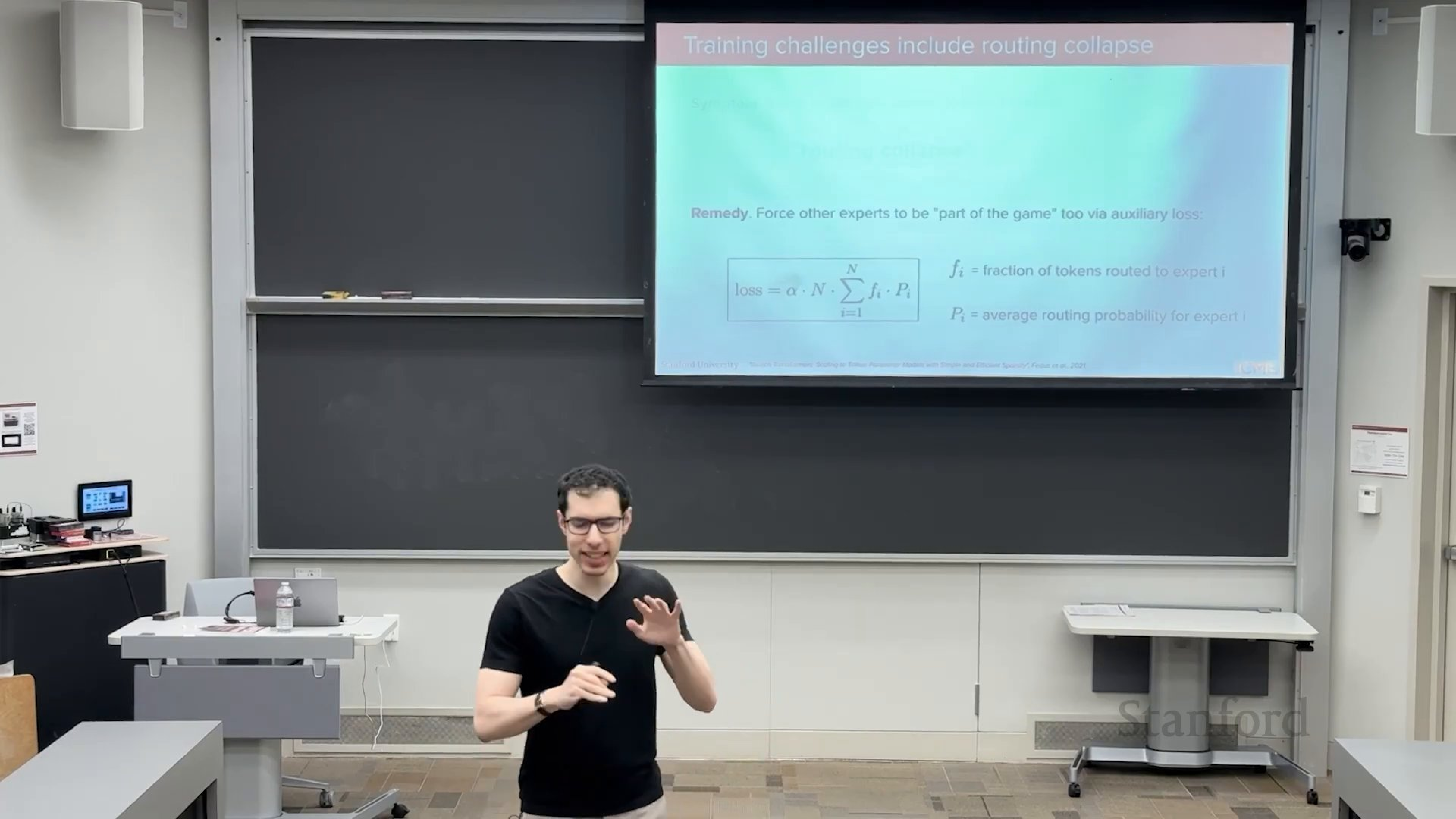

Training challenges include routing collapse
"routing collapse"
Remedy. Force other experts to be "part of the game" too via auxiliary loss:
$\text{loss} = \alpha \cdot N \cdot \sum_{i=1}^{N} f_i \cdot P_i$
f_i = fraction of tokens routed to expert i
P_i = average routing probability for expert i
Stanford University
Fedus et al, 2021
Stanford

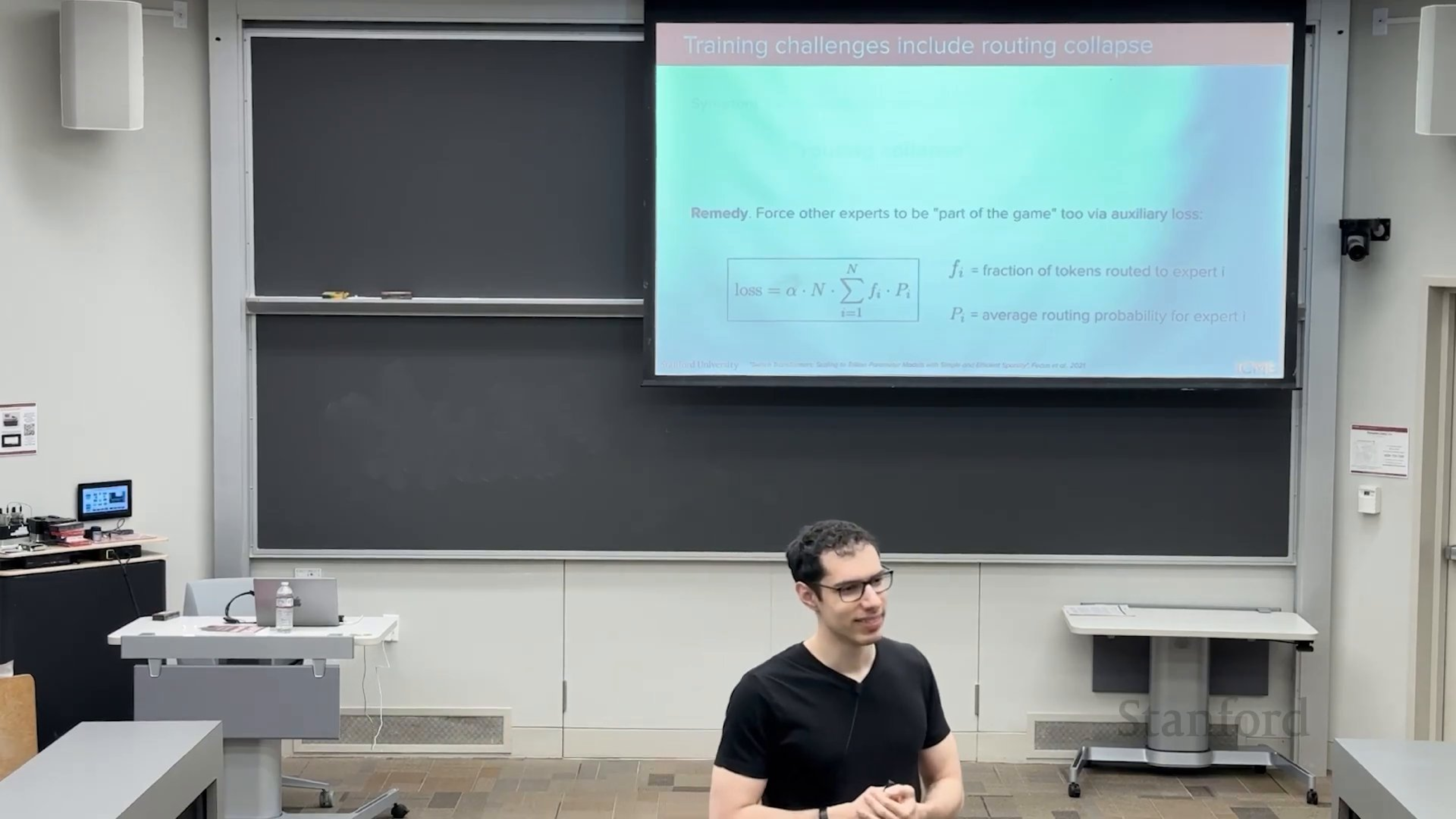


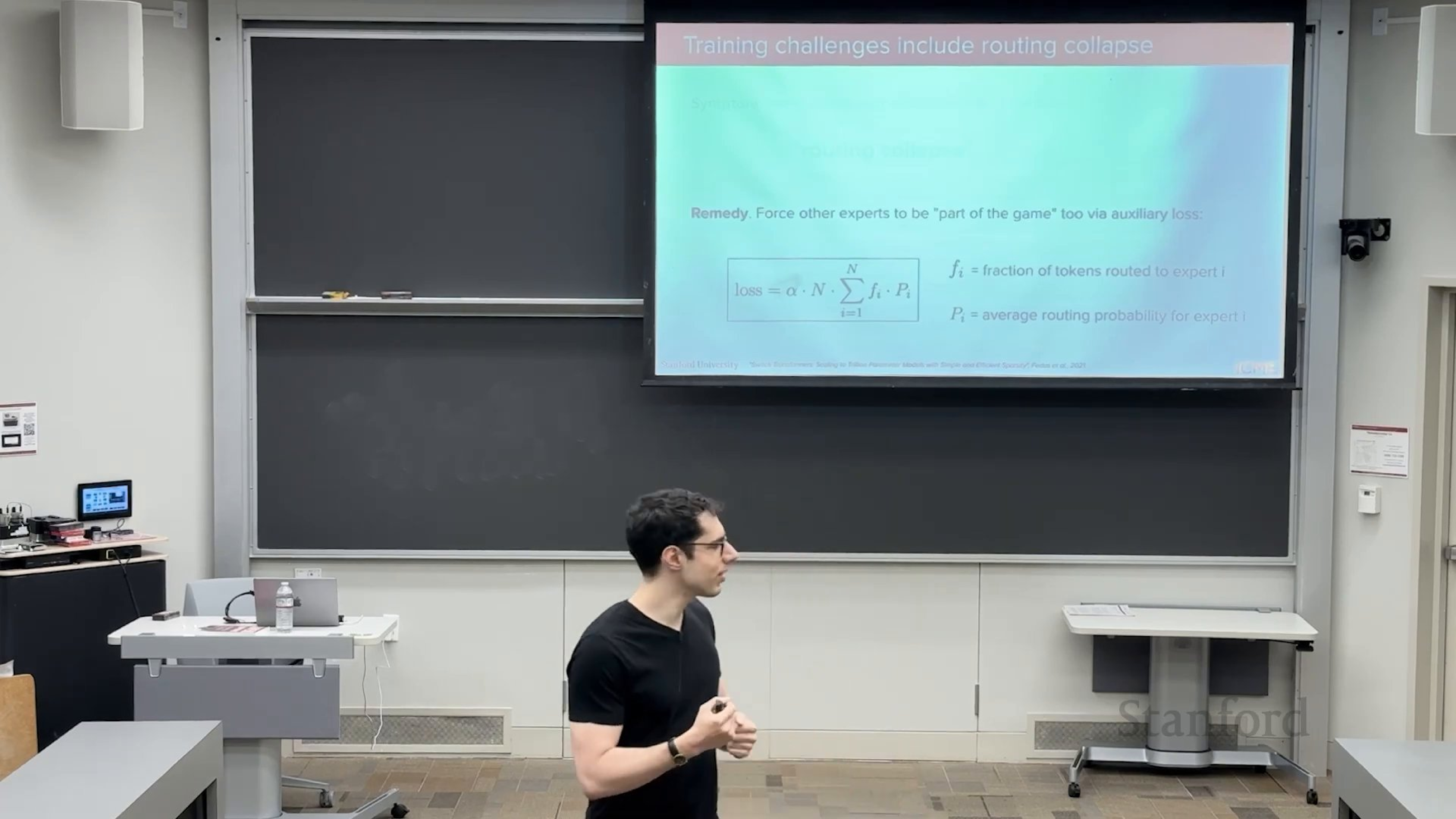
Training challenges include routing collapse
Remedy. Force other experts to be "part of the game" too via auxiliary loss:
$\text{loss} = \alpha \cdot N \cdot \sum_{i=1}^{N} f_i \cdot P_i$
f_i = fraction of tokens routed to expert i
P_i = average routing probability for expert i
Stanford

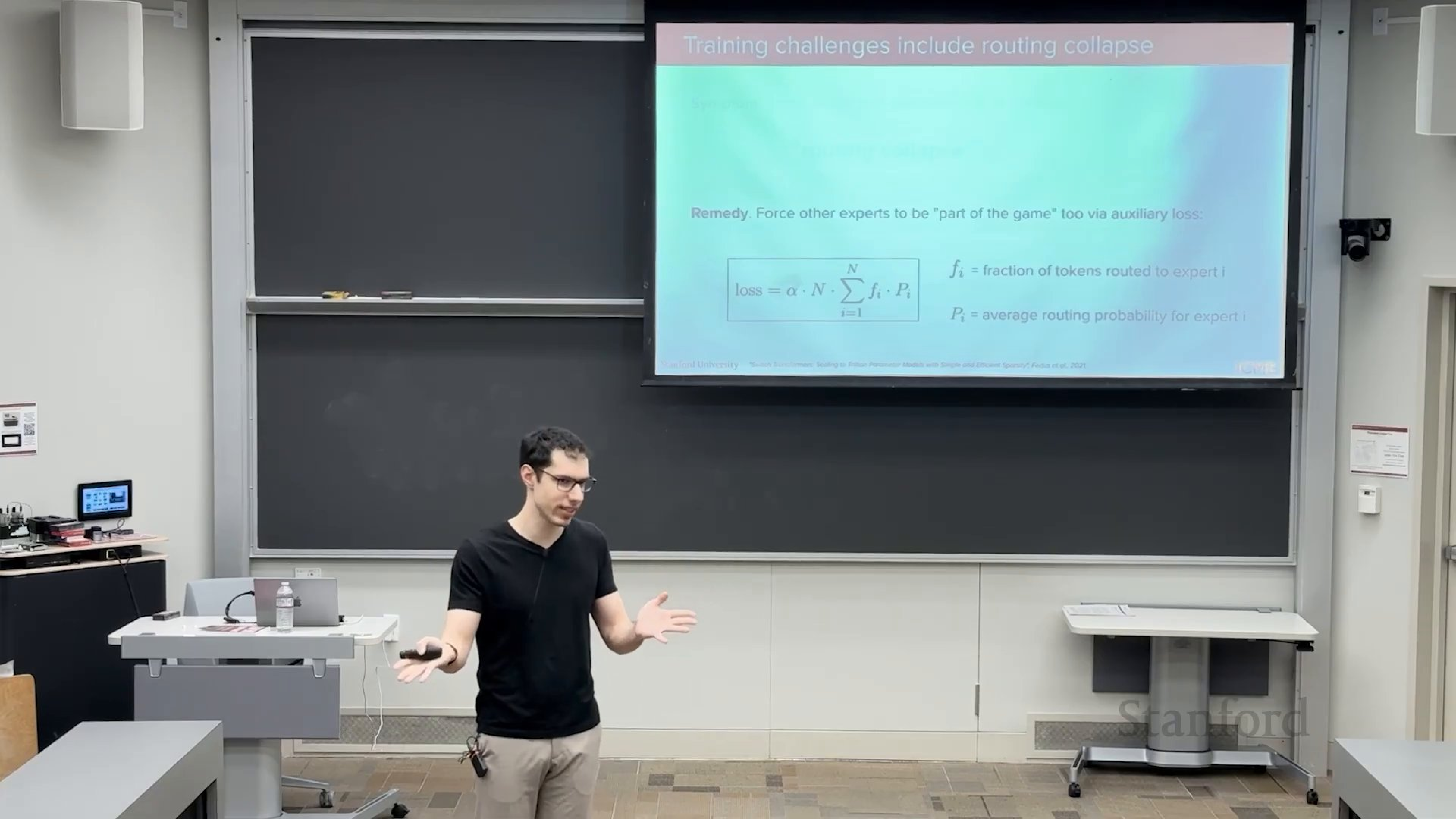


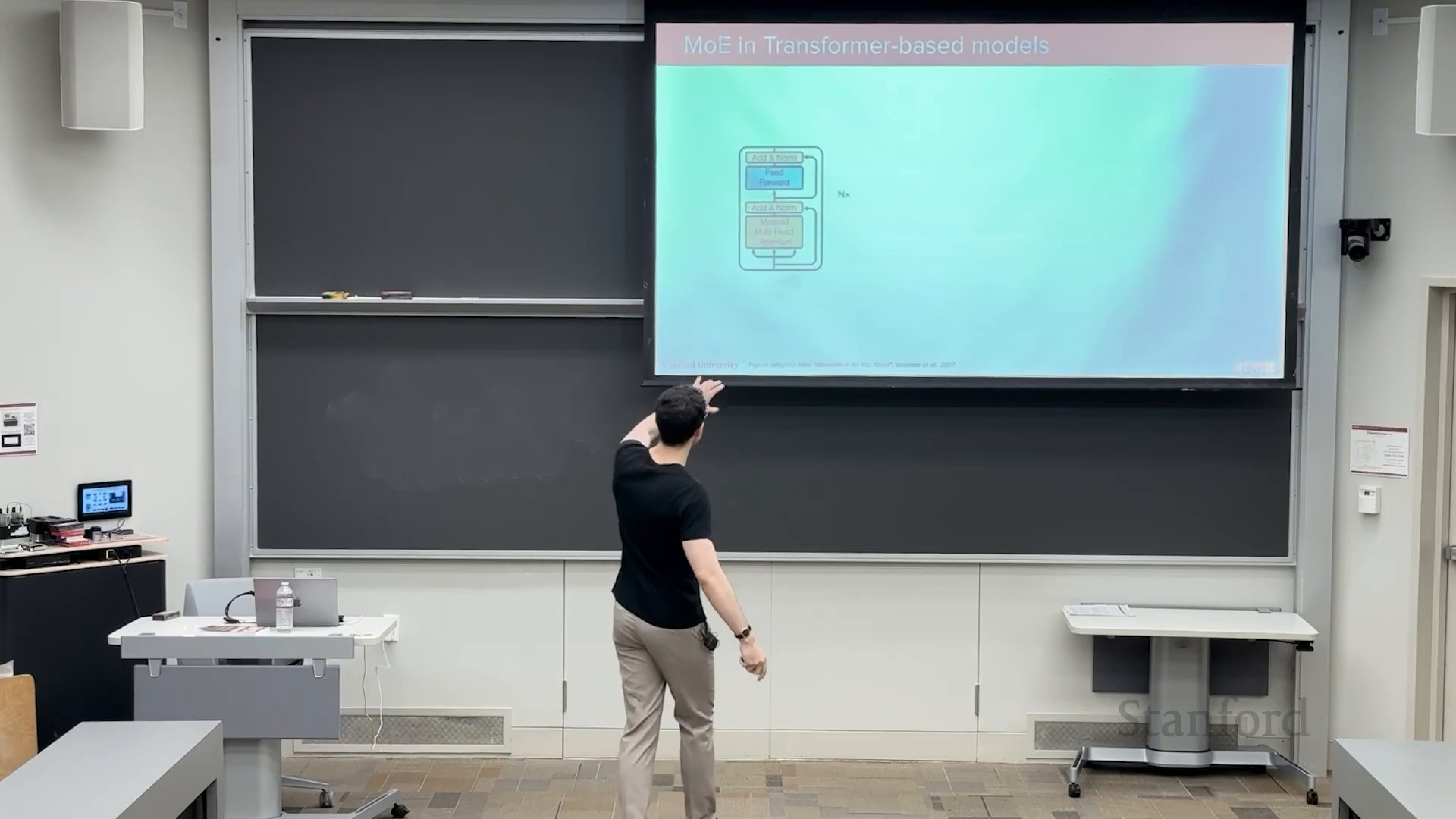
Training challenges include routing collapse
Remedy. Force other experts to be "part of the game" too via auxiliary loss:
$\boxed{\text{loss} = \alpha \cdot N \cdot \sum_{i=1}^{N} f_i \cdot P_i}$
f_i = fraction of tokens routed to expert i
P_i = average routing probability for expert i
Stanford University
"Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity", Fedus et al., 2021
Stanford


Training challenges include routing collapse
Remedy. Force other experts to be "part of the game" too via auxiliary loss:
$\boxed{\text{loss} = \alpha \cdot N \cdot \sum_{i=1}^{N} f_i \cdot P_i}$
f_i = fraction of tokens routed to expert i
P_i = average routing probability for expert i
Stanford University
"Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity", Fedus et al., 2021
ICME
Stanford


Training challenges include routing collapse
Remedy. Force other experts to be "part of the game" too via auxiliary loss:
$$\text{loss} = \alpha \cdot N \cdot \sum_{i=1}^{N} f_i \cdot P_i$$
f_i = fraction of tokens routed to expert i
P_i = average routing probability for expert i
Stanford University
"Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity", Fedus et al. 2021
Stanford


Training challenges include routing collapse
Remedy. Force other experts to be "part of the game" too via auxiliary loss:
$\text{loss} = \alpha \cdot N \cdot \sum_{i=1}^{N} f_i \cdot P_i$
f_i = fraction of tokens routed to expert i
P_i = average routing probability for expert i
Stanford University
"Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity", Fedus et al., 2021
ICME
Stanford

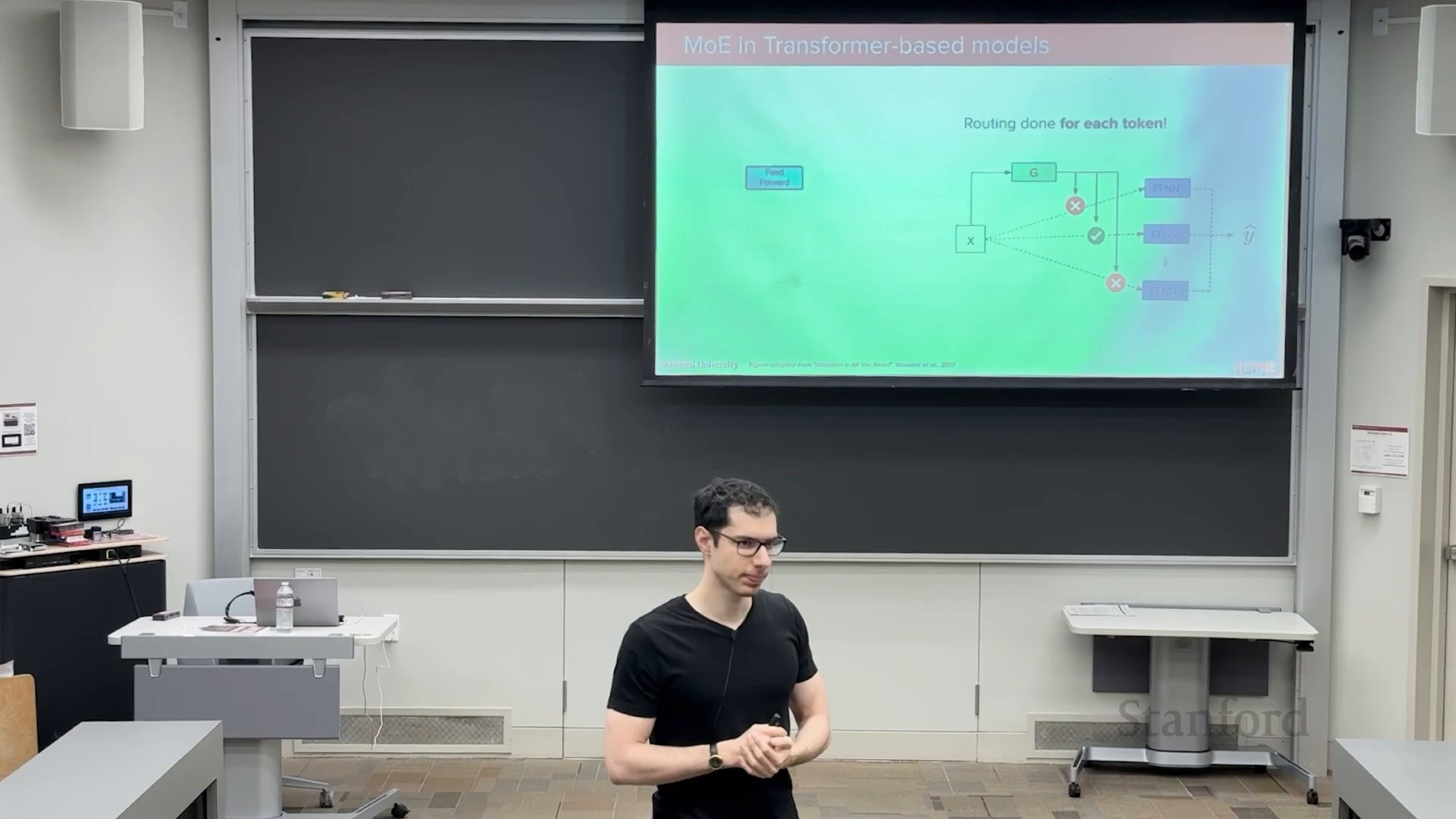

Training challenges include routing collapse
Remedy. Force other experts to be "part of the game" too via auxiliary loss:
$$\text{loss} = \alpha \cdot N \cdot \sum_{i=1}^{N} f_i \cdot P_i$$
f_i = fraction of tokens routed to expert i
P_i = average routing probability for expert i
Stanford University
"Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity", Fedus et al., 2021
Stanford


MoE in Transformer-based models
Add & Norm
Feed Forward
Add & Norm
Masked Multi Head Attention
N×
Stanford University
Figure adapted from "Attention is All You Need", Vaswani et al., 2017
Stanford


MoE in Transformer-based models
Add & Norm
Feed Forward
Add & Norm
Masked Multi-Head Attention
N×
Stanford University
Figure adapted from "Attention is All You Need" Vaswani et al. 2017
Stanford


MoE in Transformer-based models
Routing done **for each token**!
Feed Forward
G
X
FFNN
FFNN
FFNN
$\widehat{y}$
Stanford University
Stanford


MoE in Transformer-based models
Routing done **for each token**!
Feed Forward
G
x
FFNN
FFNN
FFNN
$\hat{y}$
Stanford University
Figure adapted from "Attention is All You Need" Vaswani et al., 2017
ICME
Stanford


MoE in Transformer-based models
Routing done for each token!
Feed Forward
G
X
FFNN
FFNN
FFNN
$\hat{y}$
Stanford University
Figure adapted from "Attention is All You Need", Vaswani et al., 2017
ICME
Stanford


MoE in Transformer-based models
Add & Norm
Feed Forward
Add & Norm
Masked Multi-Head Attention
Nx
Stanford University
Figure adapted from "Attention is All You Need", Vaswani et al., 2017.
Stanford


MoE in Transformer-based models
Add & Norm
Feed Forward
Add & Norm
Masked Multi-Head Attention
N×
Stanford University
Figure adapted from "Attention is All You Need", Vaswani et al., 2017
ICME
Stanford


MoE in Transformer-based models
Add & Norm
Feed Forward
Add & Norm
Masked Multi-Head Attention
N×
Stanford University
Figure adapted from "Attention is All You Need", Vaswani et al., 2017
Stanford

Interpreting experts

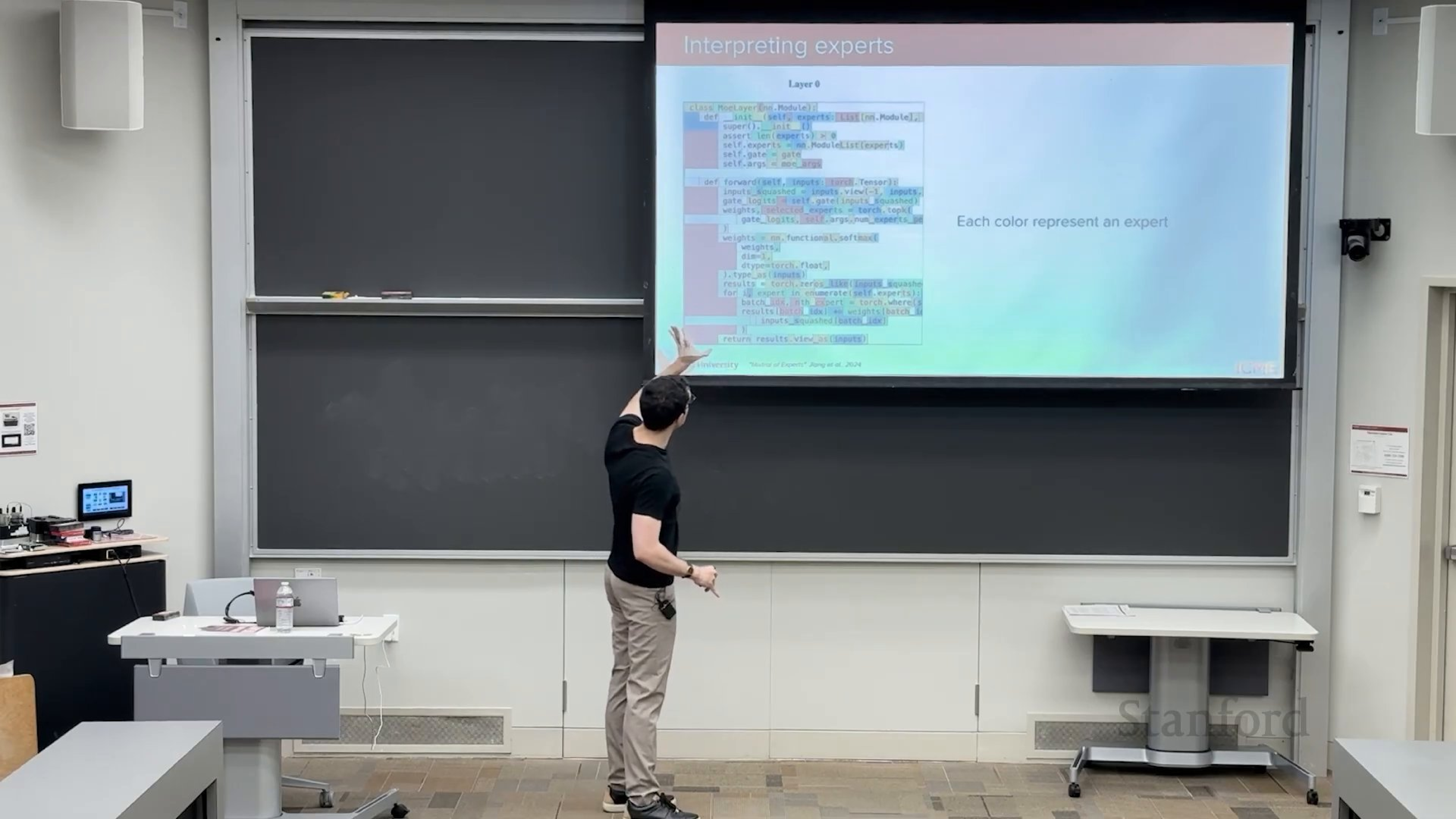
Layer 0

```
class MoeLayer(nn.Module):
    def __init__(self, experts: List[nn.Module],
        super().__init__()
        assert len(experts) > 0
        self.experts = nn.ModuleList(experts)
        self.gate = gate
        self.args = moe_args

    def forward(self, inputs: torch.Tensor):
        inputs_squashed = inputs.view(-1, inputs.
        gate_logits = self.gate(inputs_squashed)
        weights, selected_experts = torch.topk(
            gate_logits, self.args.num_experts_pe
        )
        weights = nn.functional.softmax(
            weights,
            dim=1,
            dtype=torch.float,
        ).type_as(inputs)
        results = torch.zeros_like(inputs_squashe
        for i, expert in enumerate(self.experts):
            batch_idx, nth_expert = torch.where(s
            results[batch_idx] += weights[batch_ic
                inputs_squashed[batch_idx]
            )
        return results.view_as(inputs)
```

Each color represent an expert

Stanford

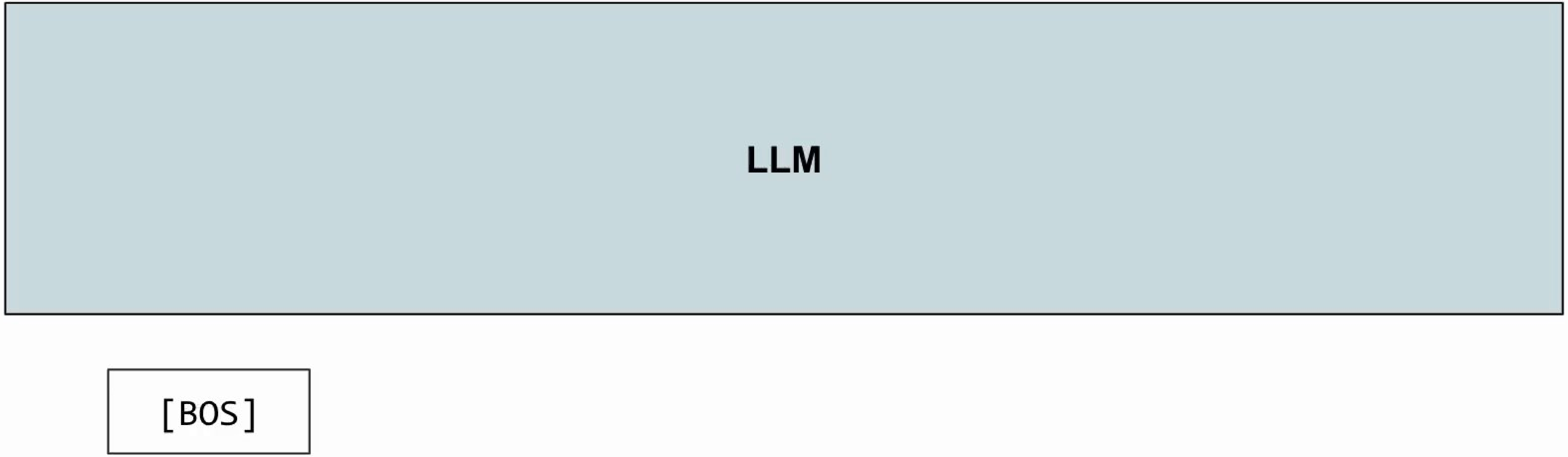


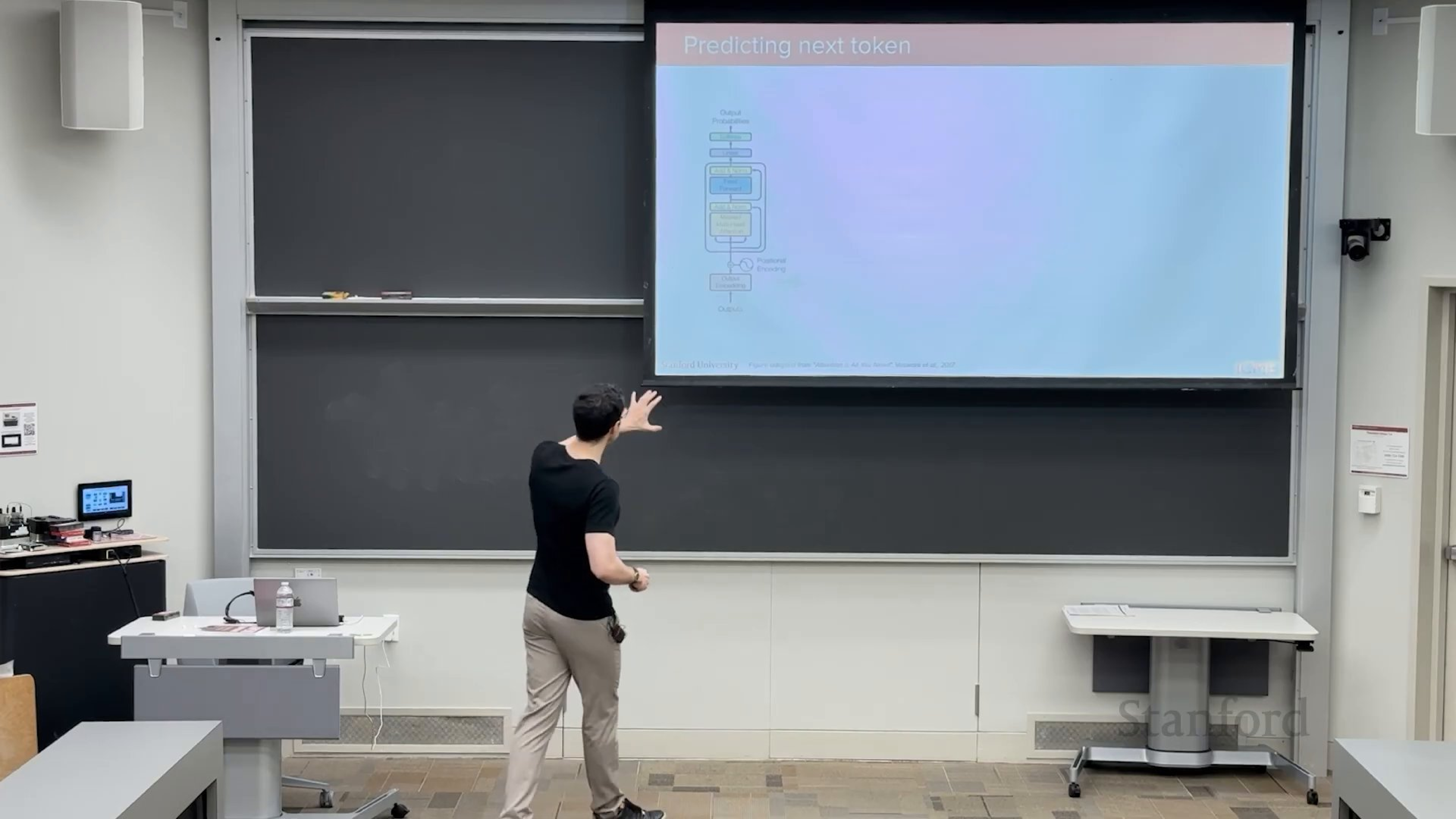
Interpreting experts
Layer 0
class MoeLayer(nn.Module):
def __init__(self, experts: List[nn.Module],
super().__init__()
assert len(experts) > 0
self.experts = nn.ModuleList(experts)
self.gate = gate
self.args = moe_args
def forward(self, inputs: torch.Tensor):
inputs_squashed = inputs.view(-1, inputs.
gate_logits = self.gate(inputs_squashed)
weights, selected_experts = torch.topk(
gate_logits, self.args.num_experts_pe
)
weights = nn.functional.softmax(
weights,
dim=1,
dtype=torch.float,
).type_as(inputs)
results = torch.zeros_like(inputs_squashe
for i, expert in enumerate(self.experts):
batch_idx, nth_expert = torch.where(s
results[batch_idx] += weights[batch_i
inputs_squashed[batch_idx]
)
return results.view_as(inputs)
Each color represent an expert
"Mixtral of Experts" Jiang et al. 2024
Stanford


Transformers & Large Language Models
LLM overview
MoE-based LLMs
Response generation
Prompting strategies
Inference optimizations
ICME
Stanford

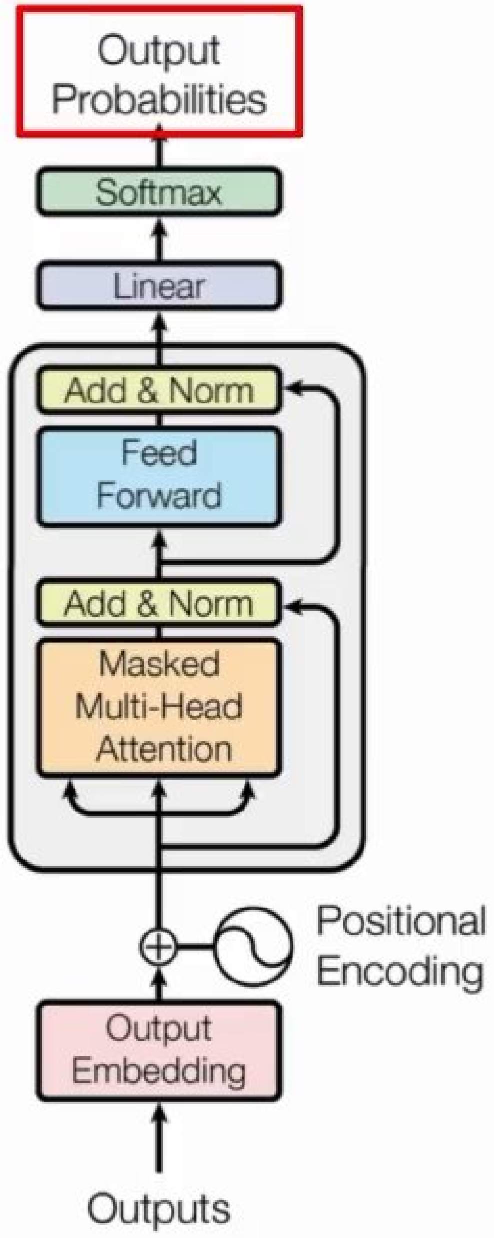

Transformers &
Large Language
Models
LLM overview
MoE-based LLMs
Response generation
Prompting strategies
Inference optimizations
ICME
Stanford

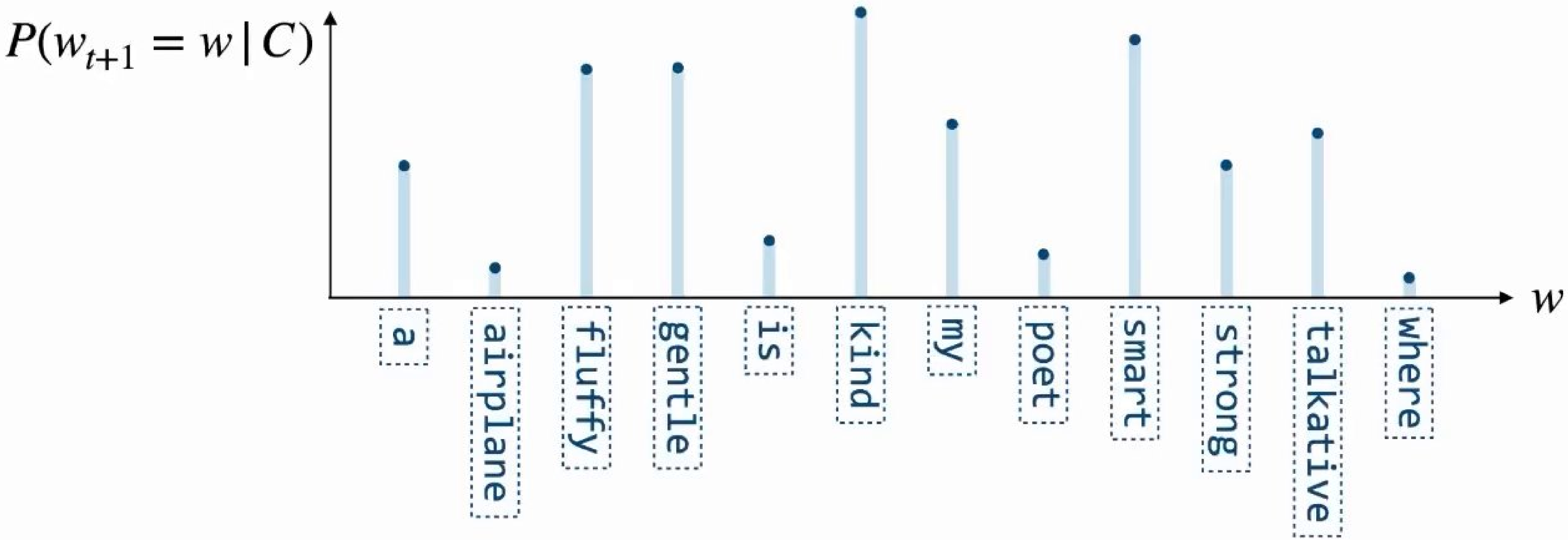
Next token prediction




Predicting next token
Output Probabilities
Add & Norm
Feed Forward
Add & Norm
Positional Encoding
Output Embedding
Stanford University
Stanford

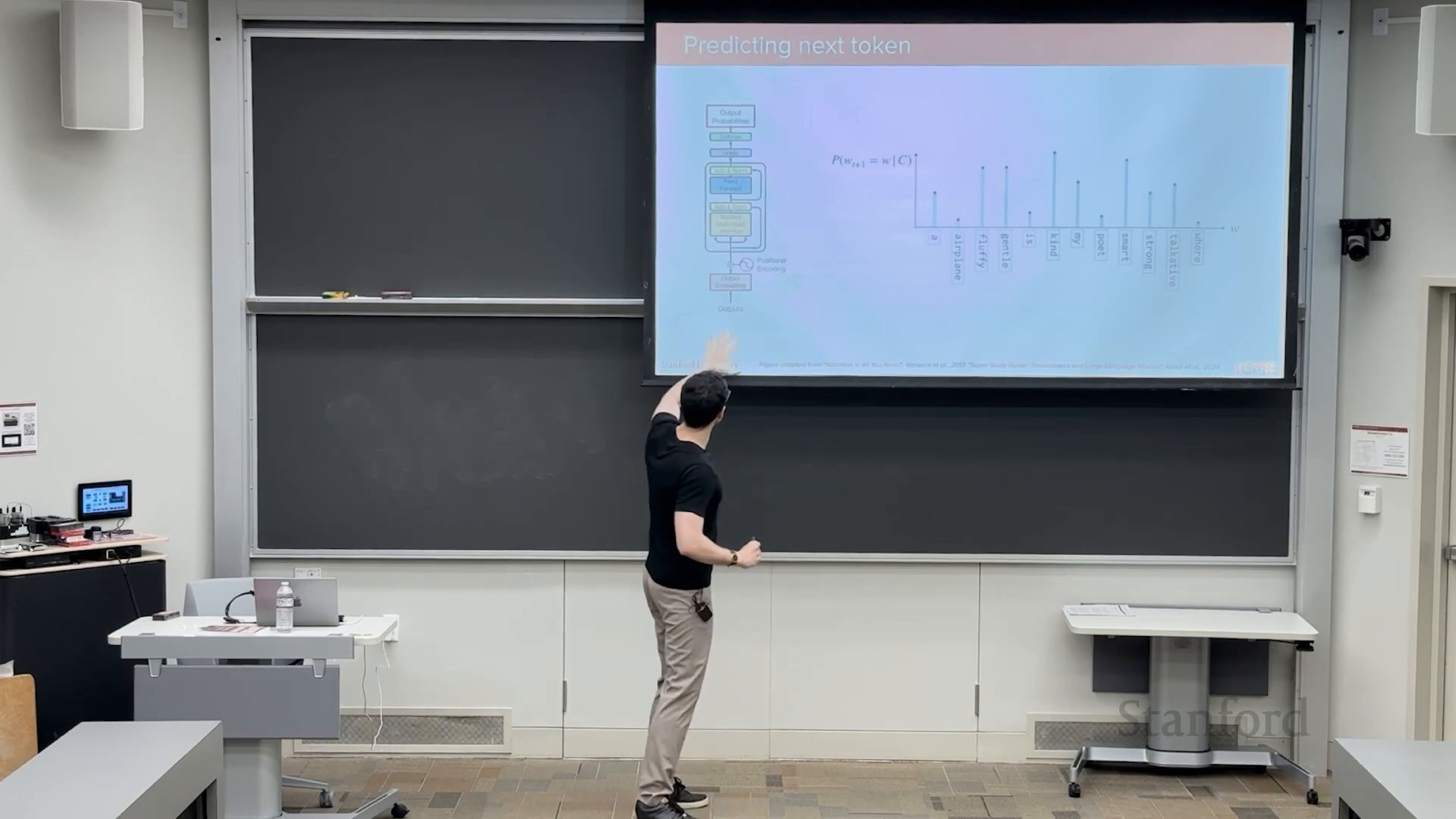
Predicting next token



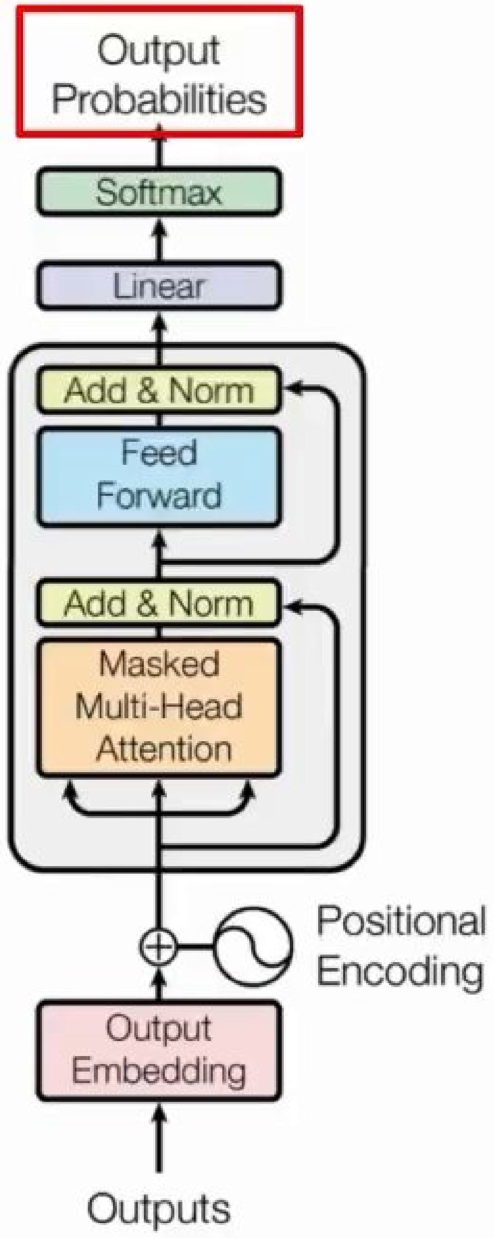


Stanford

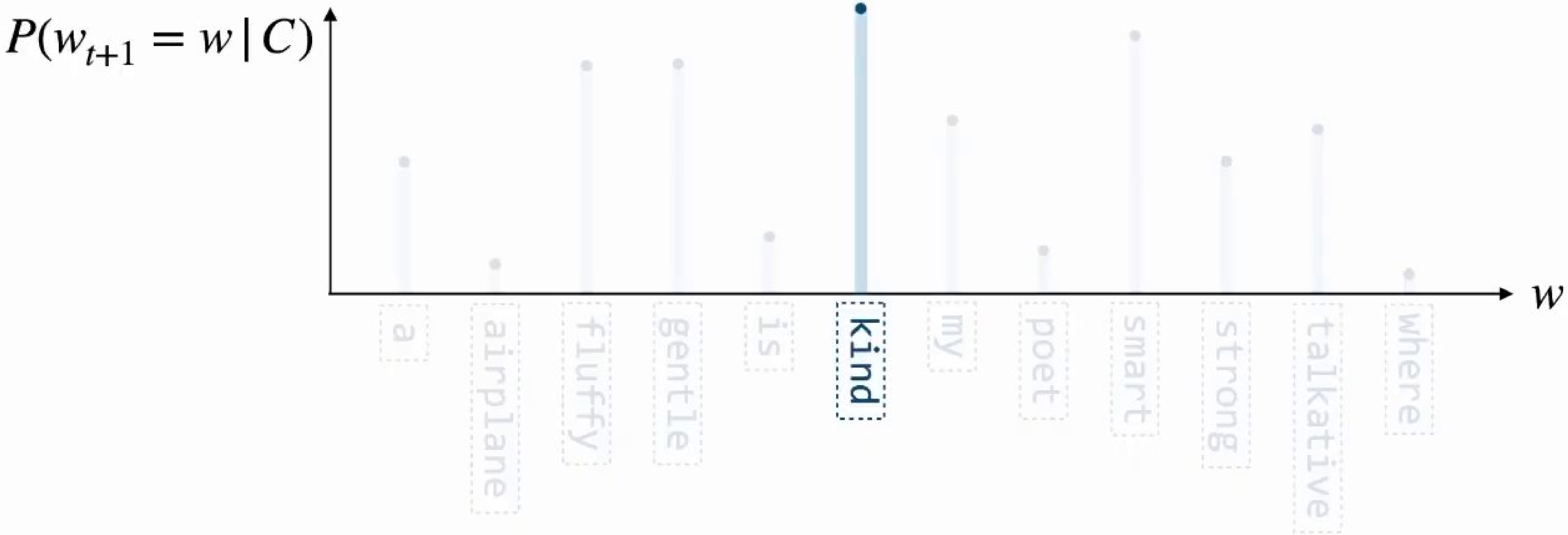
Figure adapted from "Attention is All You Need", Vaswani et al., 2017. "Super Study Guide: Transformers and Large Language Models", Amidi et al., 2024.


Predicting next token
Output Probabilities
Positional Encoding
Outputs
$P(w_{t+1} = w \mid C)$
w
a
airplane
fluffy
gentle
is
kind
my
poet
smart
strong
talkative
where
Stanford

Predicting next token with greedy decodi



1st idea. Take token with highest predicted probability



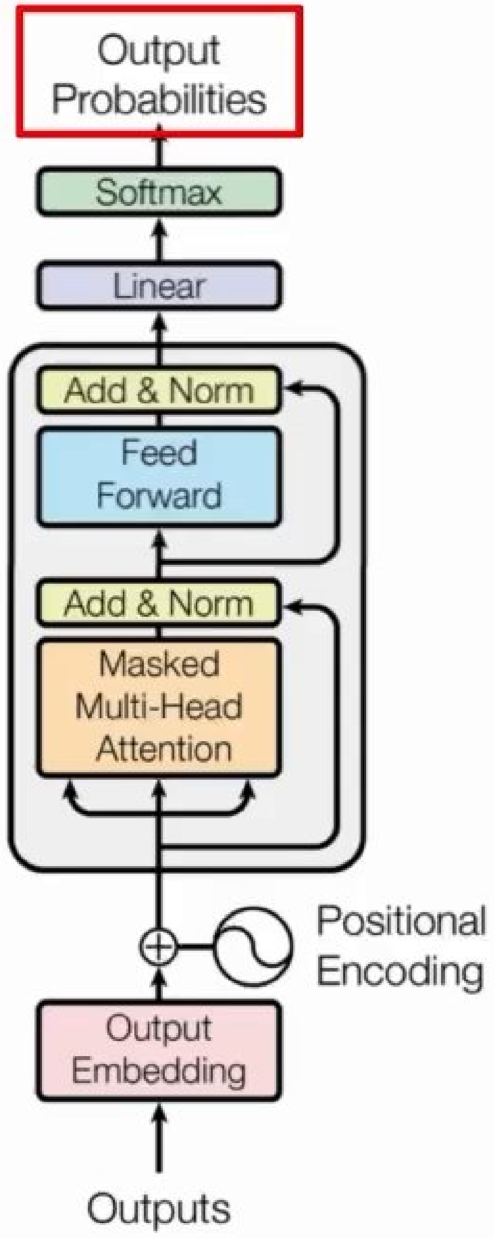
Stanford

Figure adapted from "Attention is All You Need", Vaswani et al., 2017. "Super Study Guide: Transformers and Large Language Models", Amidi et al., 2024.

Predicting next token with greedy decoding
1st idea. Take token with highest predicted probability
$P(w_{t+1} = w \mid C)$
w
k1nd
Positional Encoding
Outputs
Limitations. Output not optimal, natural and/or diverse
Stanford University
Stanford


Predicting next token with greedy decoding
1st idea. Take token with highest predicted probability
Output Probabilities
Positional Encoding
Output Embedding
Outputs
$P(w_{t+1} = w \mid C)$
w
kind
Limitations. Output not optimal, natural and/or diverse
Stanford University
Stanford

Predicting next token with beam search



2nd idea. Keep k paths that are the most likely

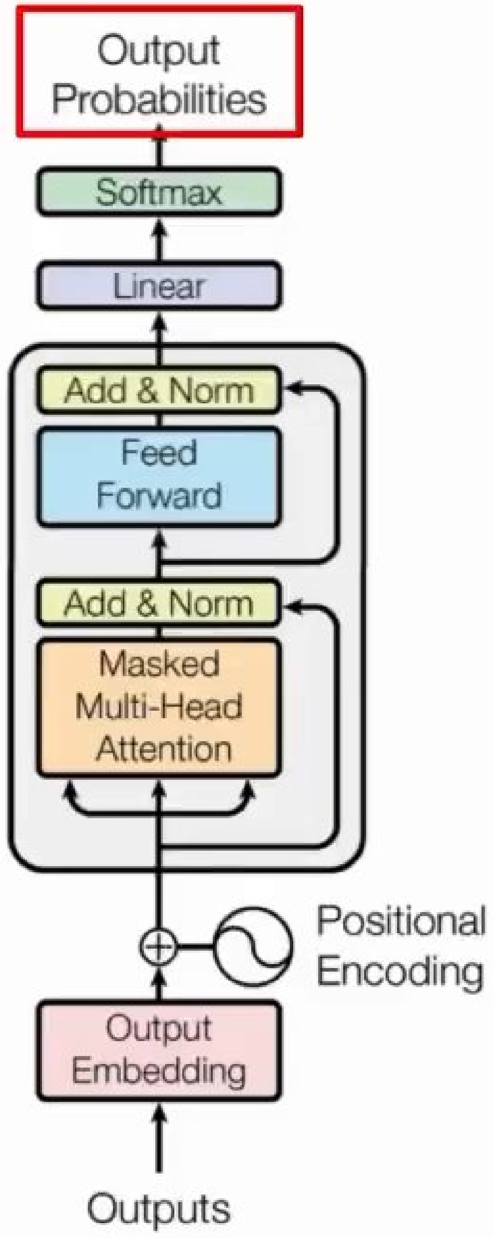
[BOS]

Stanford

Figure adapted from "Attention is All You Need", Vaswani et al., 2017.

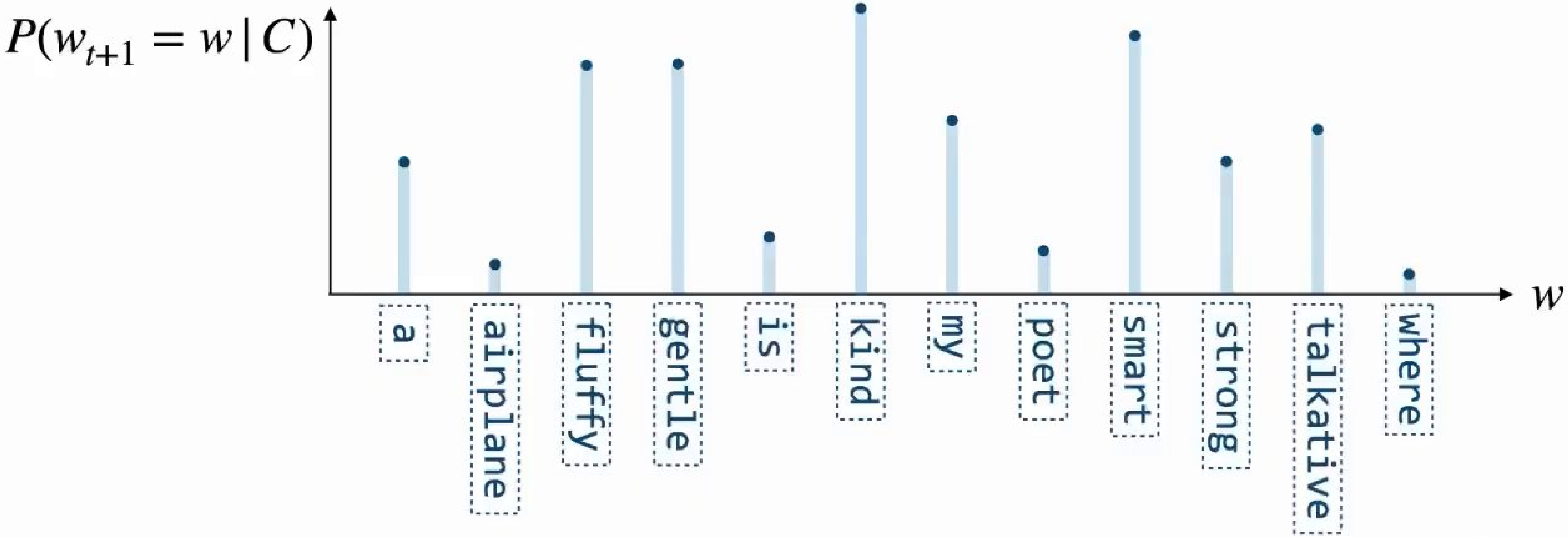
Predicting next token with beam search
2nd idea. Keep k paths that are the most likely
Output Probabilities
Positional Encoding
Outputs
[BOS]
a
cute
teddy
bear
Ends at [EOS]
Stanford University
Stanford

Predicting next token with sampling



3rd idea. Sample next token from probability distribution:

$$\boxed{\widehat{w}_{t+1} \sim P(w_{t+1}|C)}$$



"Super Study Guide: Transformers and Large Language Models", Amidi et al., 2024.

Predicting next token with sampling
3rd idea. Sample next token from probability distribution:
$\widehat{w}_{t+1} \sim P(w_{t+1}|C)$
$P(w_{t+1} = w \mid C)$
w
a
airplane
fluffy
gentle
is
kind
my
poet
smart
strong
talkative
where
Output Probabilities
Positional Encoding
Output Embedding
Outputs
Stanford University
"Super Study Guide: Transformers and Large Language Models", Amidi et al., 2024
Stanford

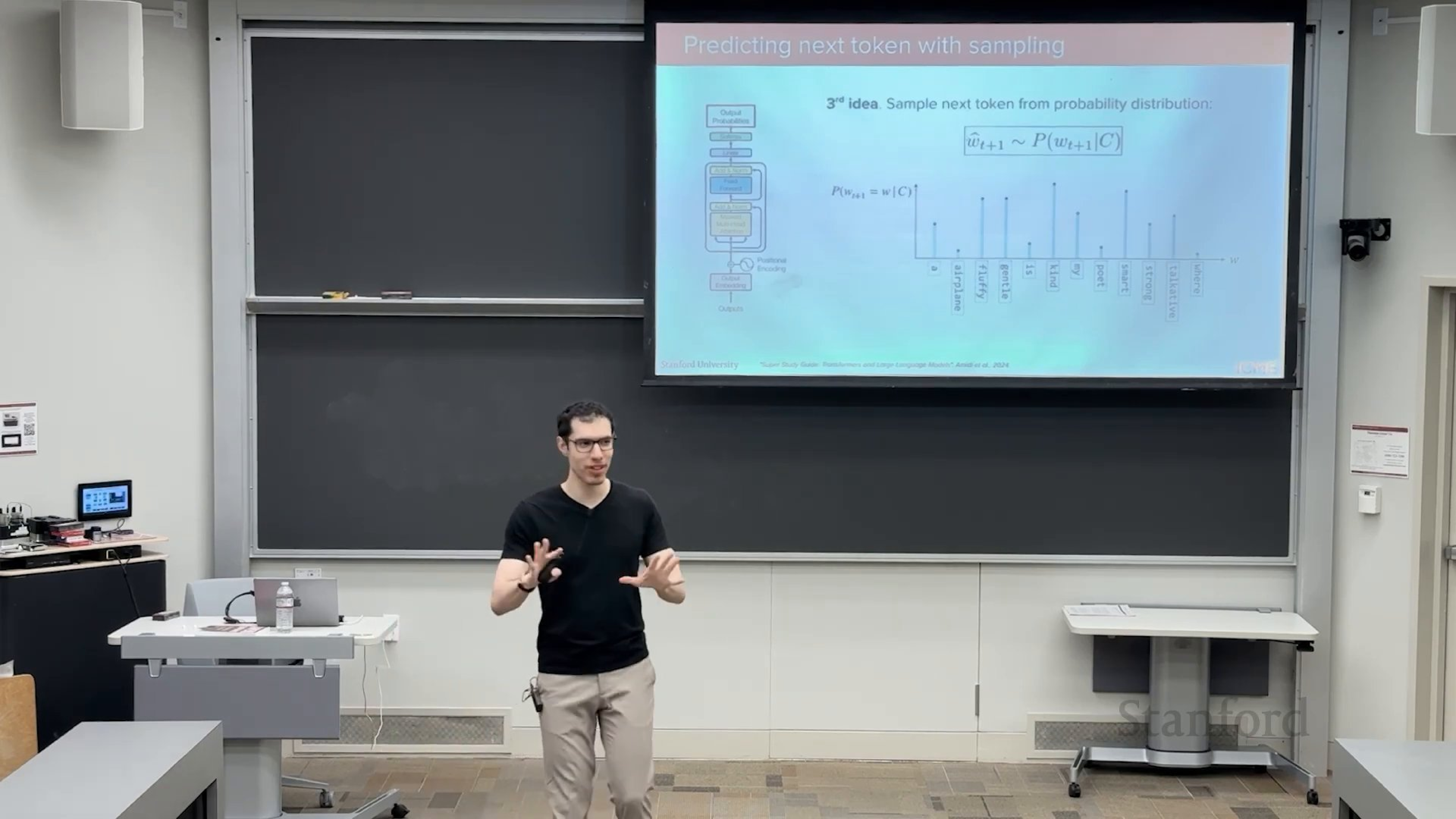
Predicting next token with sampling
3rd idea. Sample next token from probability distribution:
$\boxed{\widehat{w}_{t+1} \sim P(w_{t+1}|C)}$
Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Masked Multi-Head Attention
Positional Encoding
Output Embedding
Outputs
$P(w_{t+1} = w\,|\,C)$
a
airplane
fluffy
gentle
is
kind
my
poet
smart
strong
talkative
where
w
Stanford University
"Super Study Guide: Transformers and Large Language Models", Amidi et al., 2024
ICME
Stanford

Predicting next token with sampling
3rd idea. Sample next token from probability distribution:
$\widehat{w}_{t+1} \sim P(w_{t+1}|C)$
$P(w_{t+1} = w \,|\, C)$
w
a
airplane
fluffy
gentle
is
kind
my
poet
smart
strong
talkative
where
Output Probabilities
Positional Encoding
Outputs
Stanford University
"Super Study Guide: Transformers and Large Language Models", Amidi et al., 2024
Stanford


Predicting next token with sampling
3rd idea. Sample next token from probability distribution:
$\boxed{\widehat{w}_{t+1} \sim P(w_{t+1}|C)}$
$P(w_{t+1} = w\,|\,C)$
w
a
airplane
fluffy
gentle
is
kind
my
poet
smart
strong
talkative
where
Output Probabilities
Positional Encoding
Output Embedding
Outputs
Stanford University
"Super Study Guide: Transformers and Large Language Models", Amidi et al., 2024
Stanford

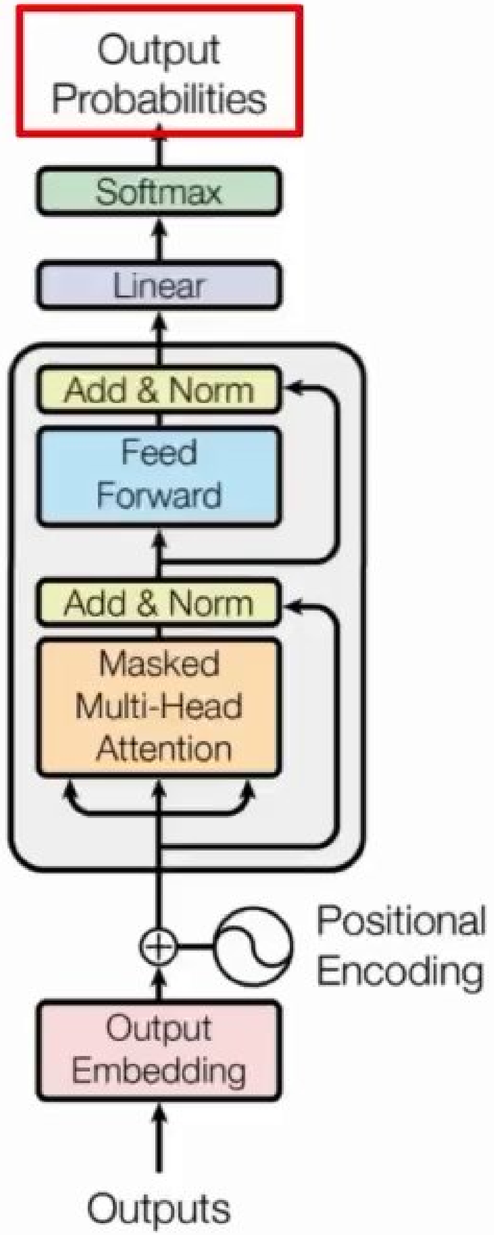

Predicting next token with sampling
3rd idea. Sample next token from probability distribution:
$\widehat{w}_{t+1} \sim P(w_{t+1}|C)$
$P(w_{t+1} = w|C)$
a
airplane
fluffy
gentle
is
kind
my
poet
smart
strong
talkative
where
w
Output Probabilities
Positional Encoding
Output Embedding
Outputs
Stanford University
"Super Study Guide: Transformers and Large Language Models", Amidi et al., 2024
ICME
Stanford

Predicting next token with sampling
3rd idea. Sample next token from probability distribution:
$\widehat{w}_{t+1} \sim P(w_{t+1}|C)$
$P(w_{t+1} = w\,|\,C)$
w
a
airplane
fluffy
gentle
is
kind
my
poet
smart
strong
talkative
where
Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Masked Multi-Head Attention
Positional Encoding
Output Embedding
Outputs
Stanford University
"Super Study Guide: Transformers and Large Language Models", Amidi et al., 2024
Stanford



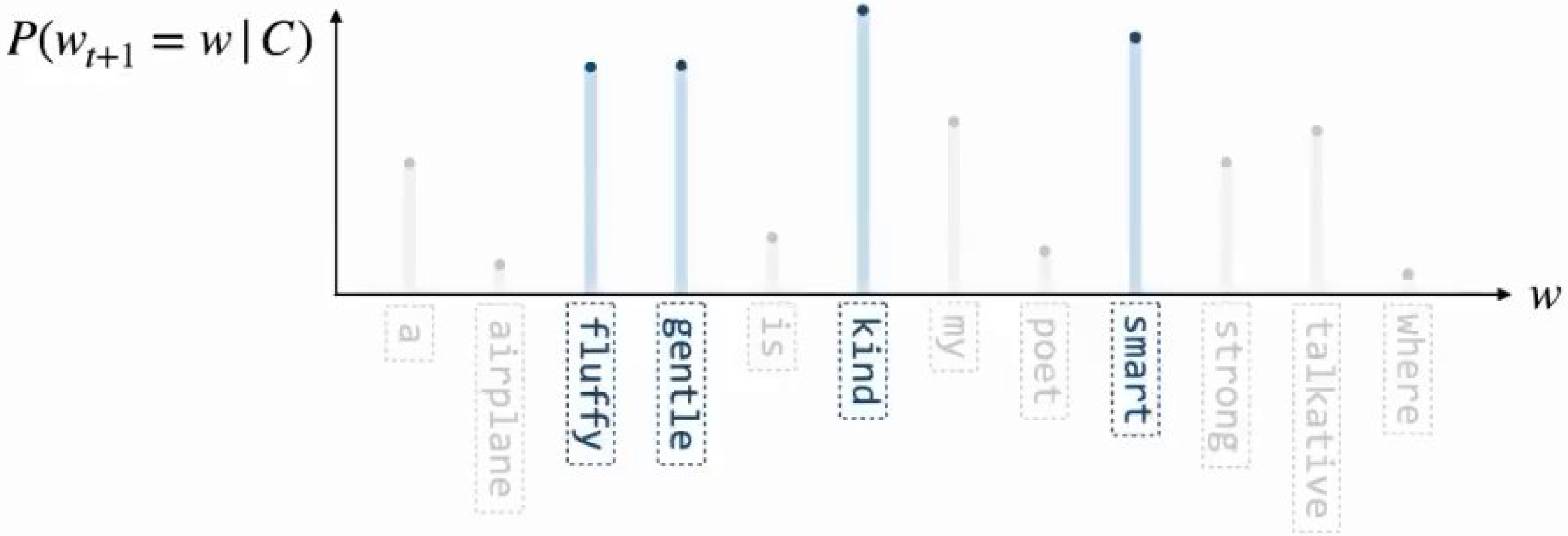
Predicting next token with sampling
3rd idea. Sample next token from probability distribution:
$\widehat{w}_{t+1} \sim P(w_{t+1}|C)$
Output Probabilities
Positional Encoding
Output Embedding
Outputs
$P(w_{t+1} = w|C)$
a
airplane
fluffy
gentle
is
kind
my
poet
smart
strong
talkative
where
w
Stanford University
"Super Study Guide: Transformers and Large Language Models", Amidi et al., 2024
ICME
Stanford

Sampling strategies



- **Top-k**: Sample among top k most probable tokens

k = 4

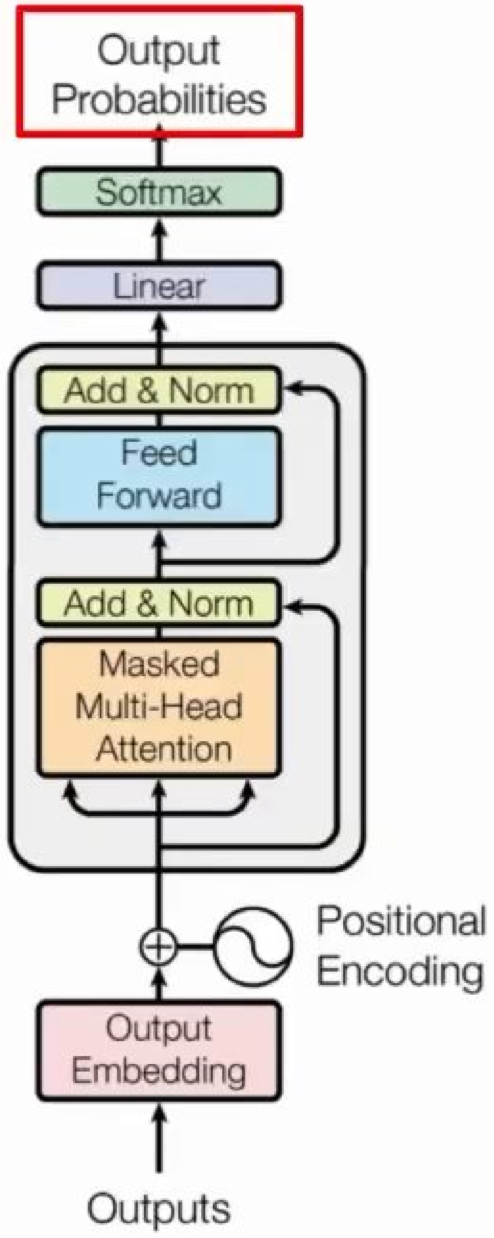


Stanford

Next token prediction

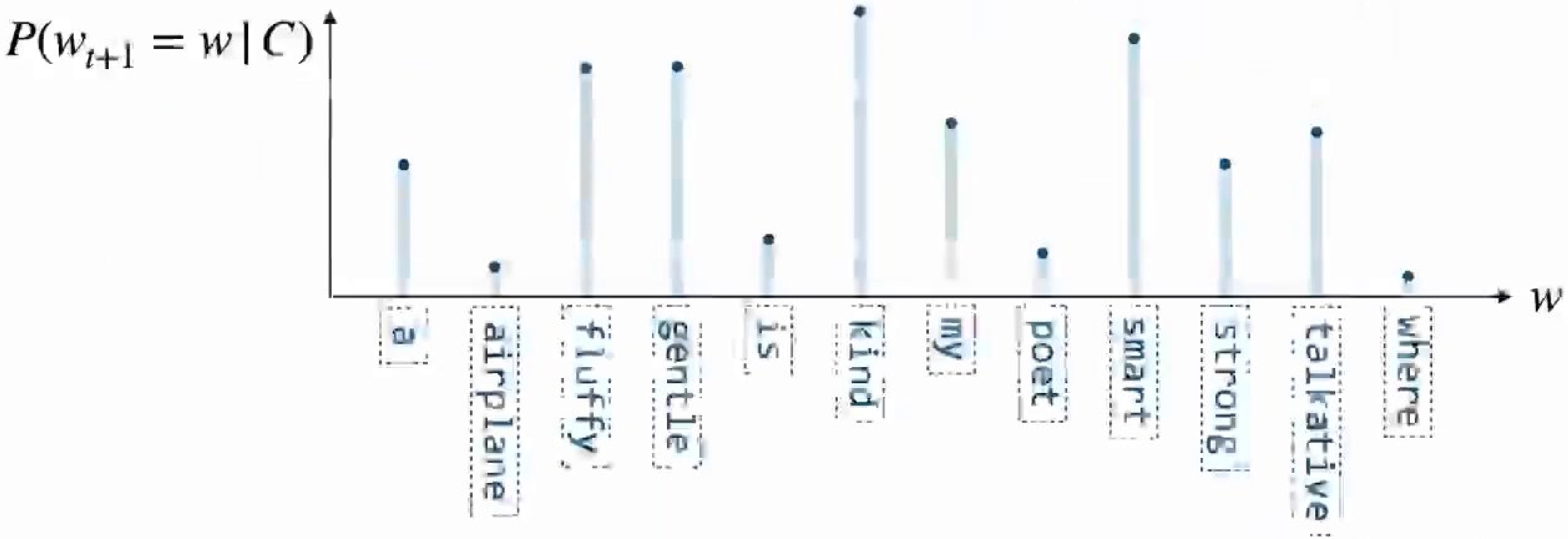
But **how** are probabilities **obtained**?

Predicting next token with sampling



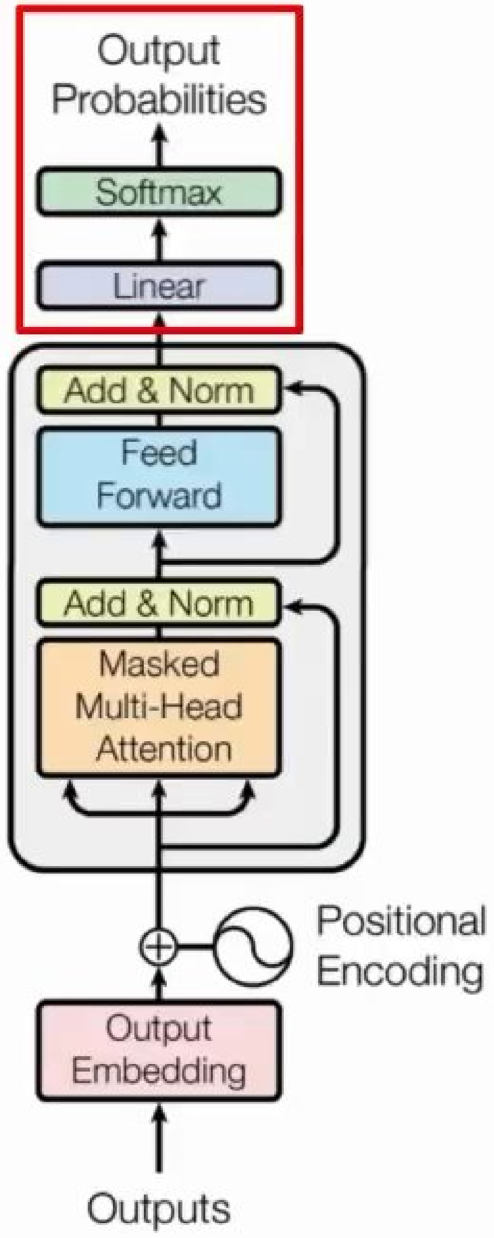
3rd idea. Sample next token from probability distribution:

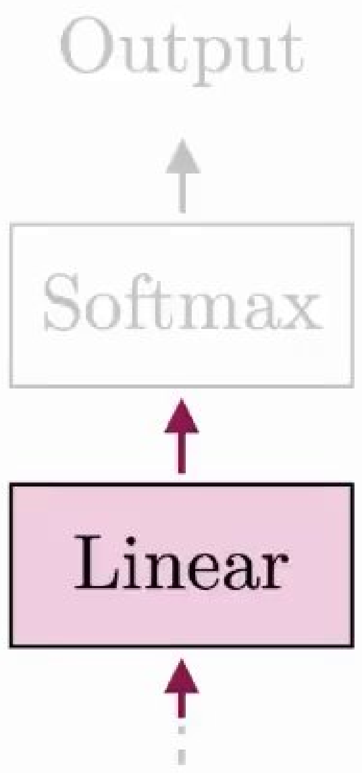
$$\boxed{\widehat{w}_{t+1} \sim P(w_{t+1}|C)}$$

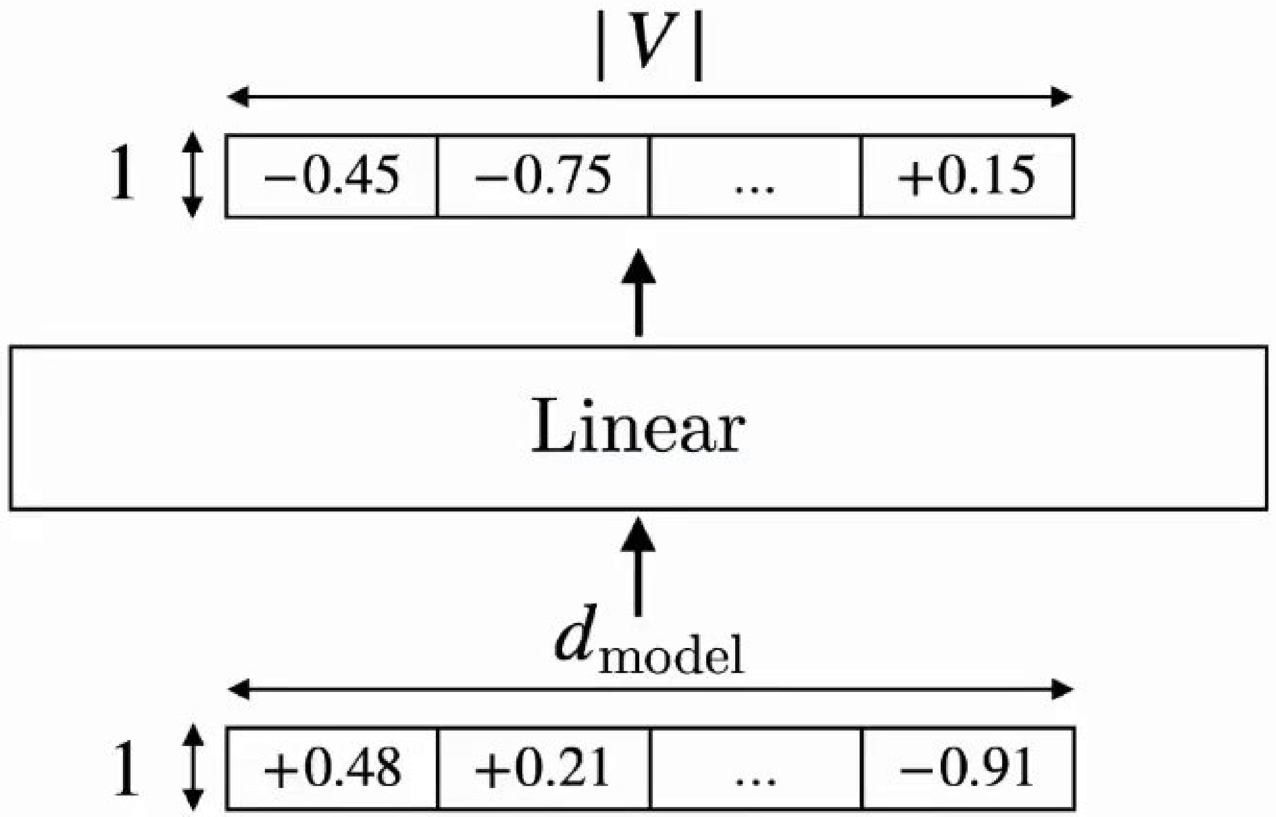


Stanford

Probability computation



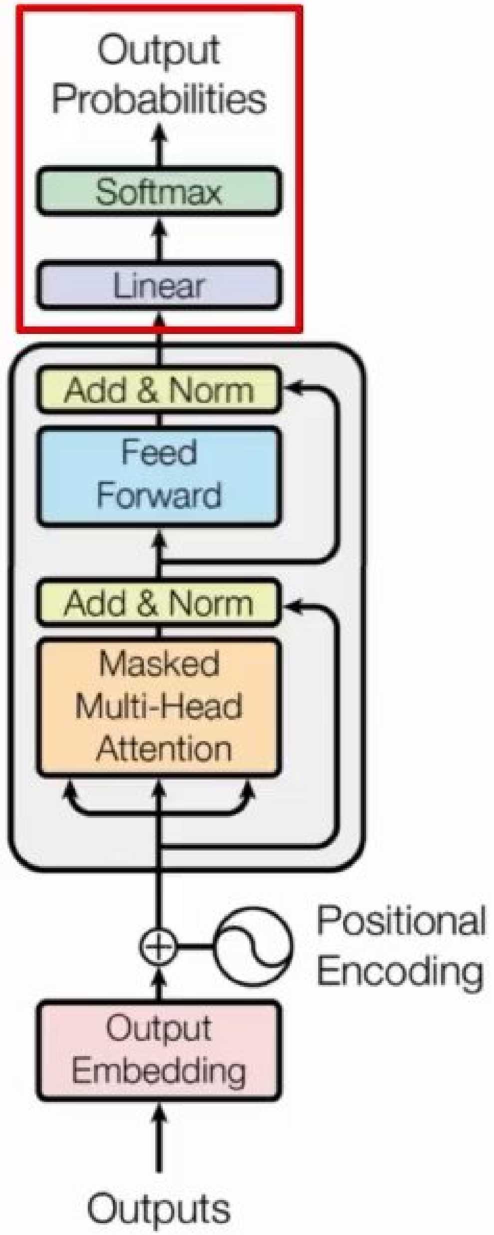


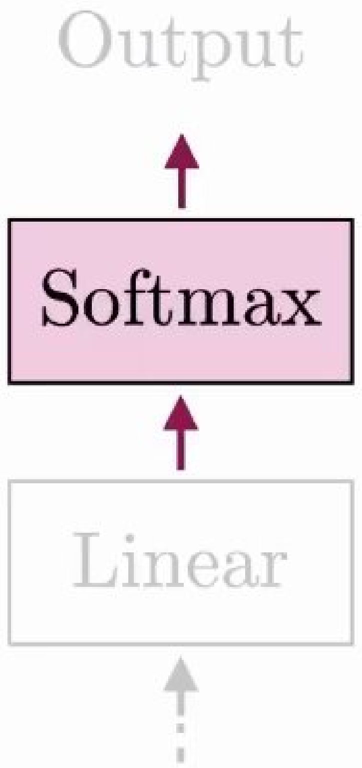


Stanford


Probability computation
Output Probabilities
Positional Encoding
Outputs
Linear
|V|
1
−0.45
−0.75
...
+0.15
Linear
d_model
1
+0.48
+0.21
...
−0.91
Stanford

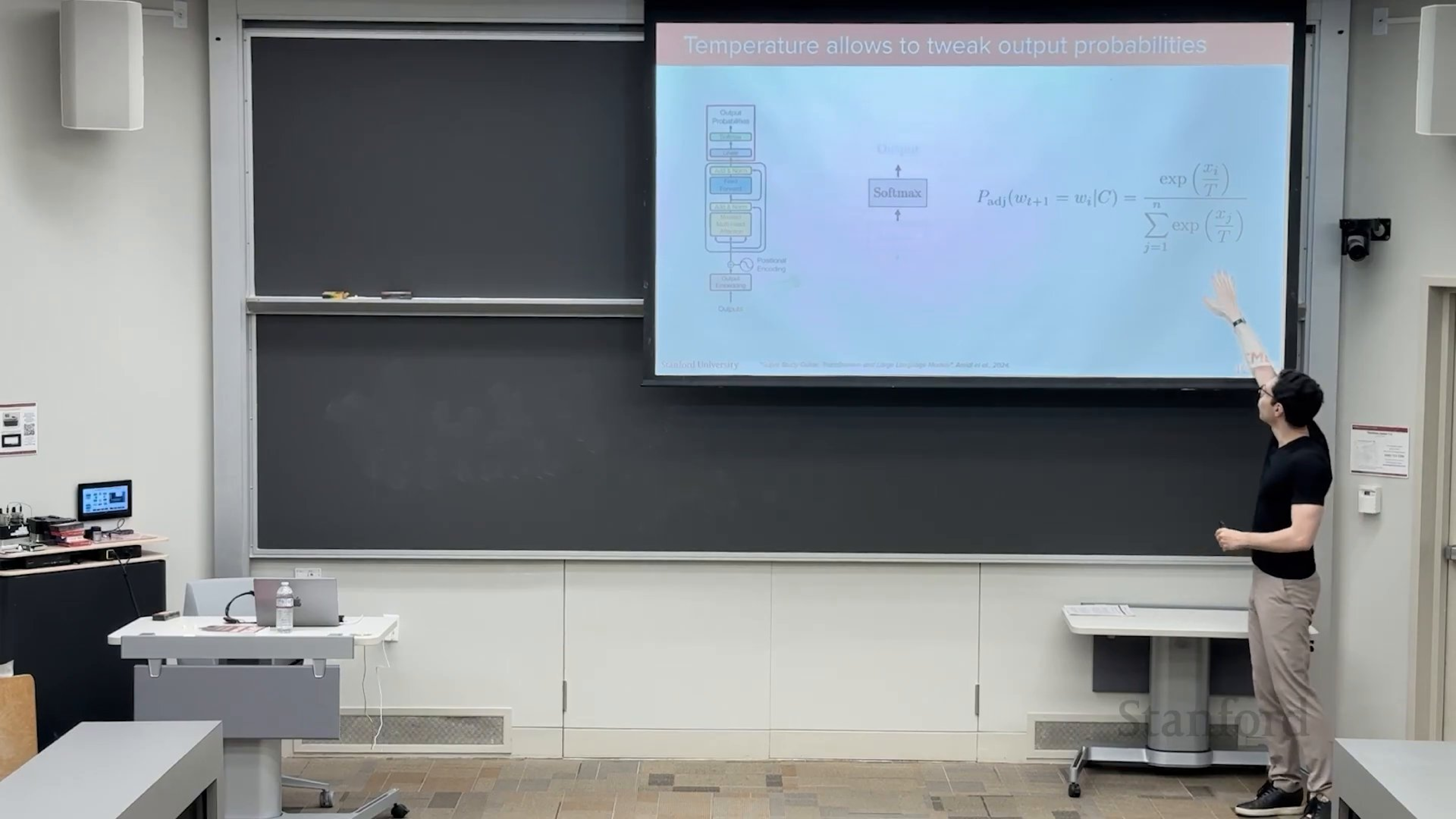
Temperature allows to tweak output prob



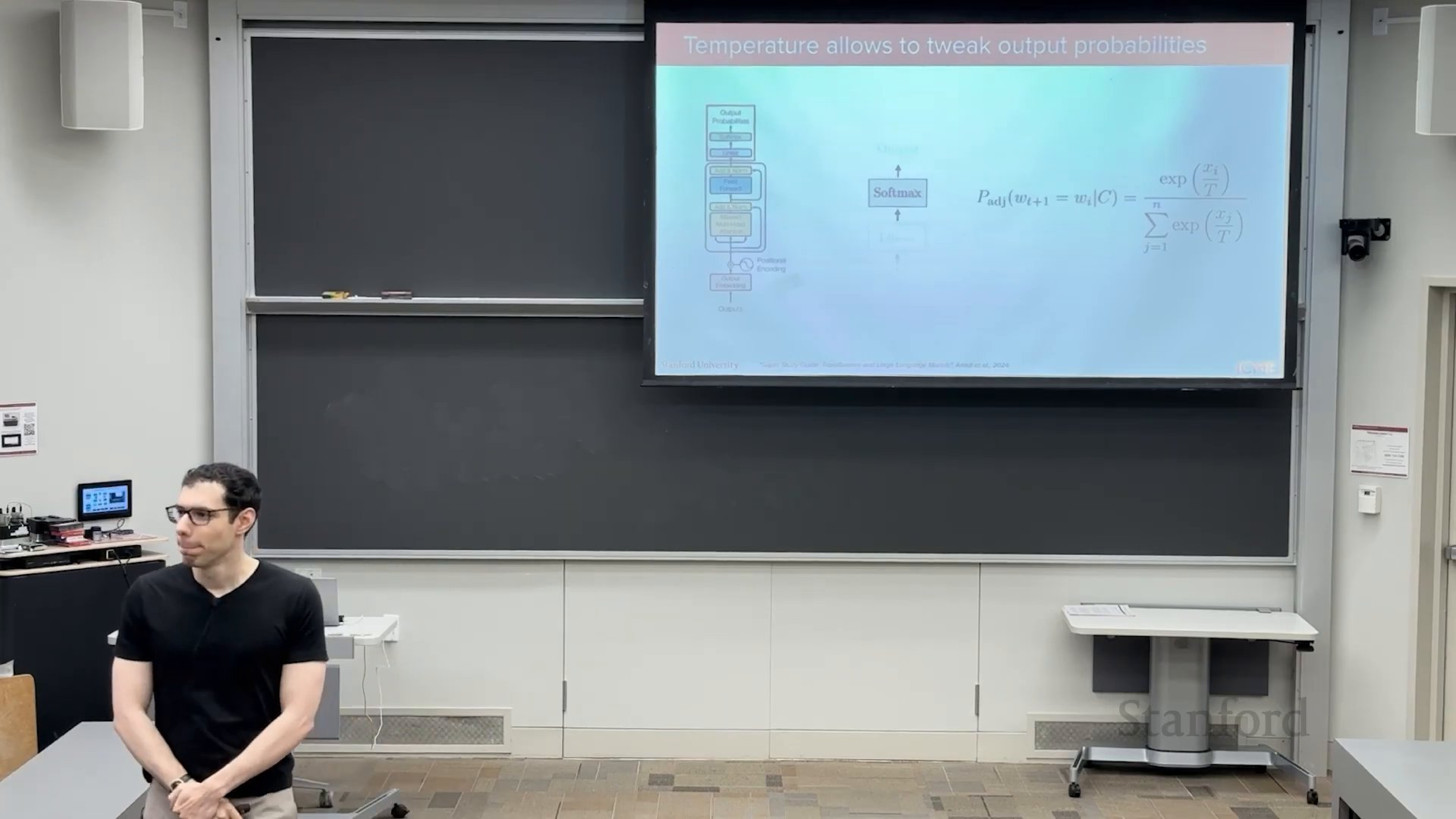


$$P_{\text{adj}}(w_{t+1} = w_i | C) = \frac{\exp\left(\frac{x_i}{T}\right)}{\sum_{j=1}^{n} \exp\left(\frac{x_j}{T}\right)}$$

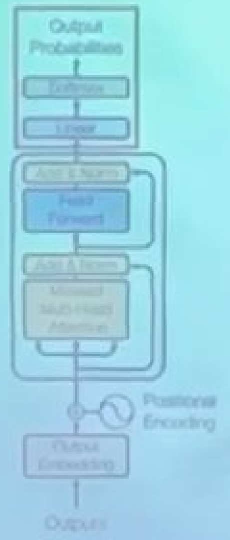
Stanford



Temperature allows to tweak output probabilities
Output Probabilities
Positional Encoding
Output Embedding
Outputs
Softmax
$$P_{\mathrm{adj}}(w_{t+1} = w_i | C) = \frac{\exp\left(\frac{x_i}{T}\right)}{\sum_{j=1}^{n} \exp\left(\frac{x_j}{T}\right)}$$
Stanford University
Stanford

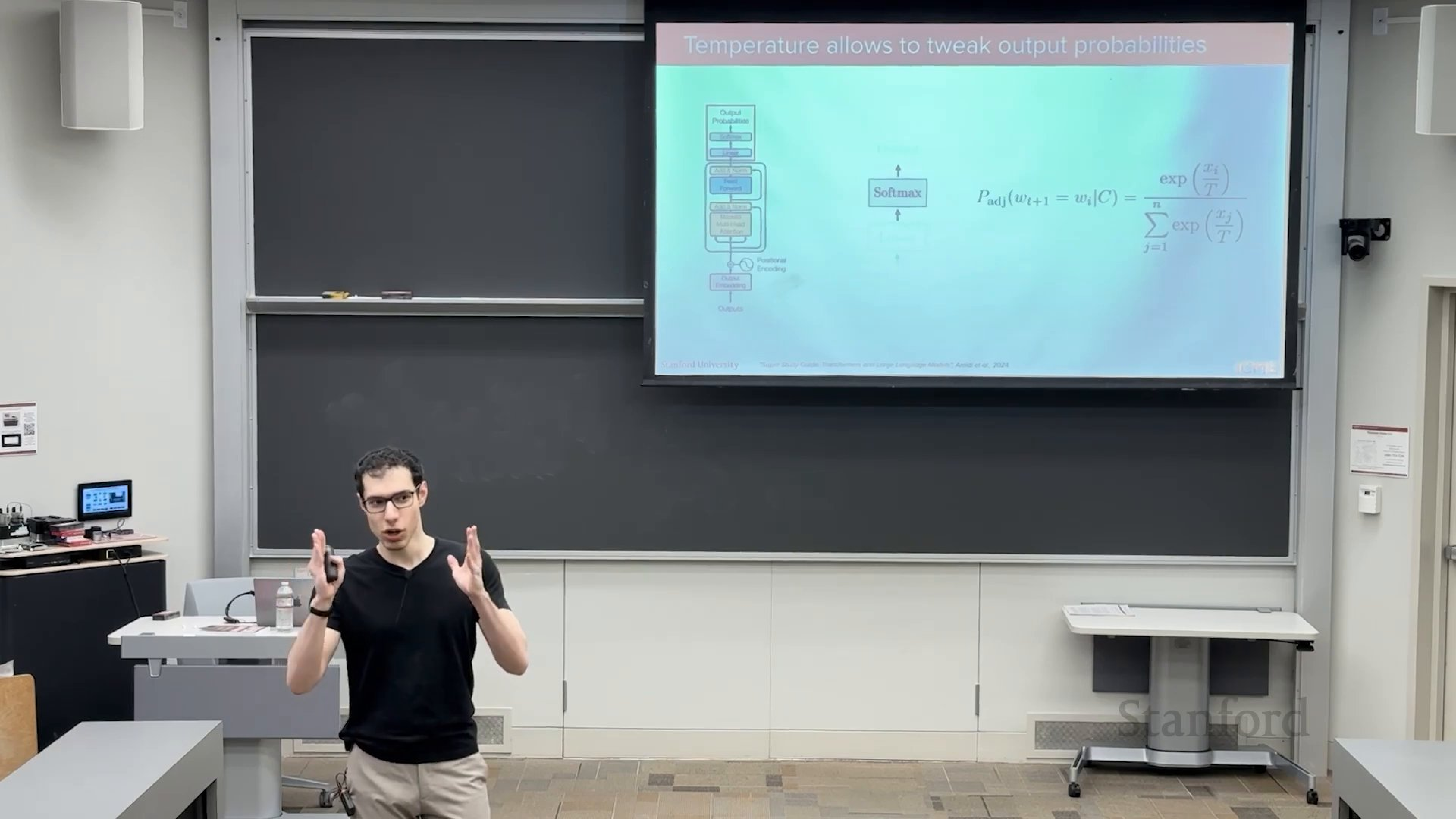

Temperature allows to tweak output probabilities
Output Probabilities
Positional Encoding
Outputs
Softmax
$P_{\text{adj}}(w_{t+1} = w_i | C) = \dfrac{\exp\left(\frac{x_i}{T}\right)}{\sum_{j=1}^{n} \exp\left(\frac{x_j}{T}\right)}$
Stanford University
Amidi et al., 2024
Stanford

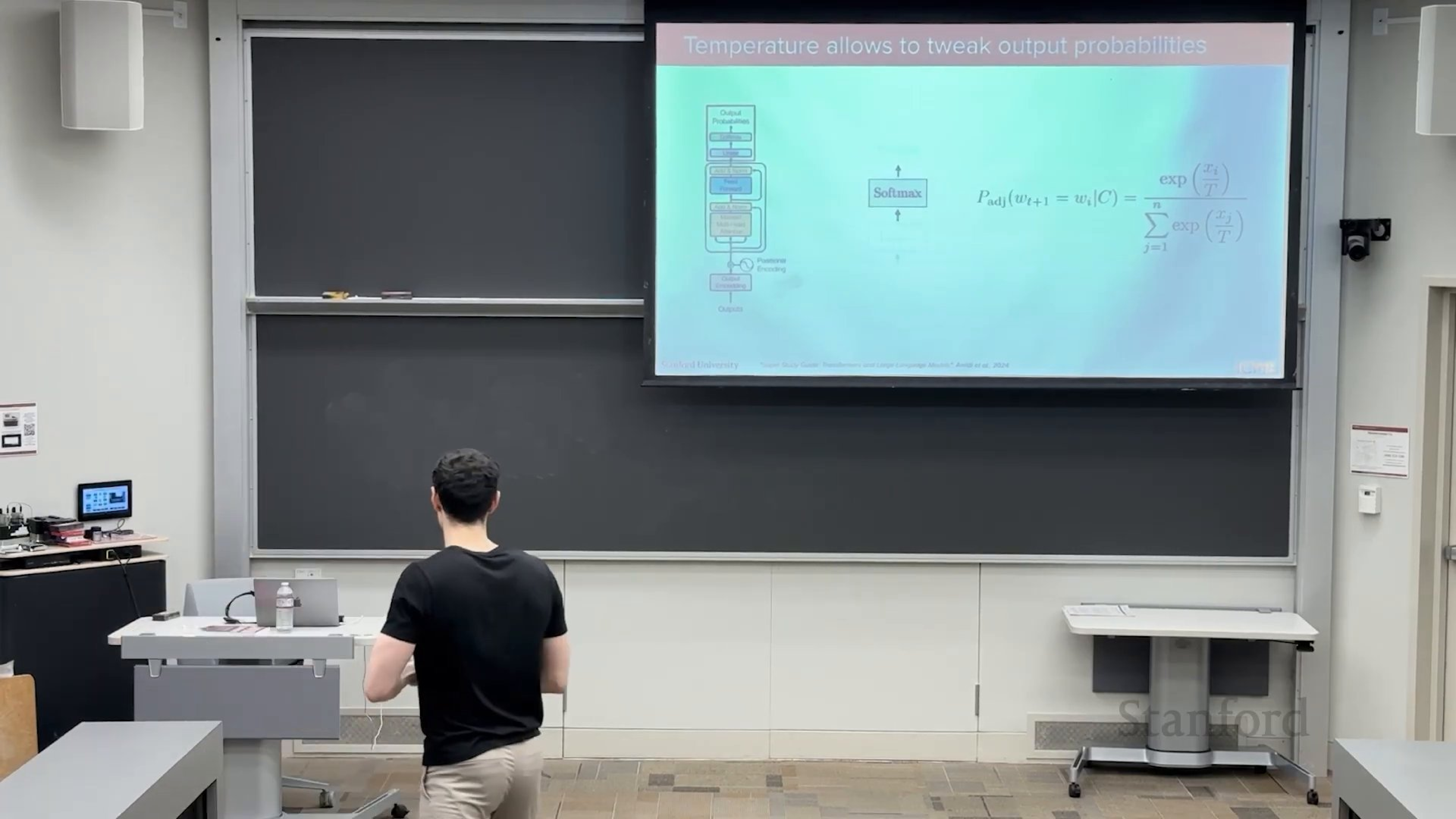
Temperature allows to tweak output probabilities

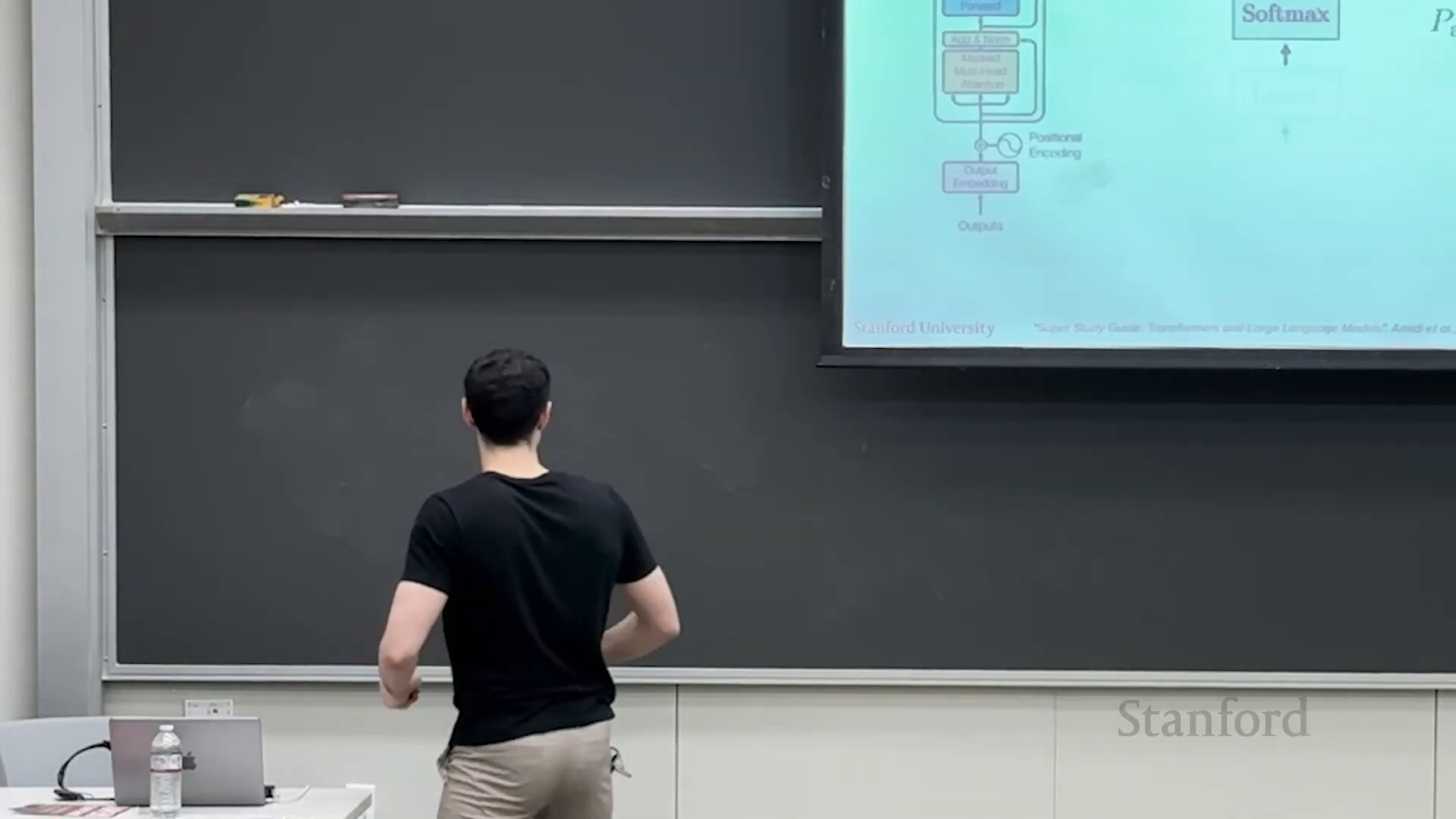


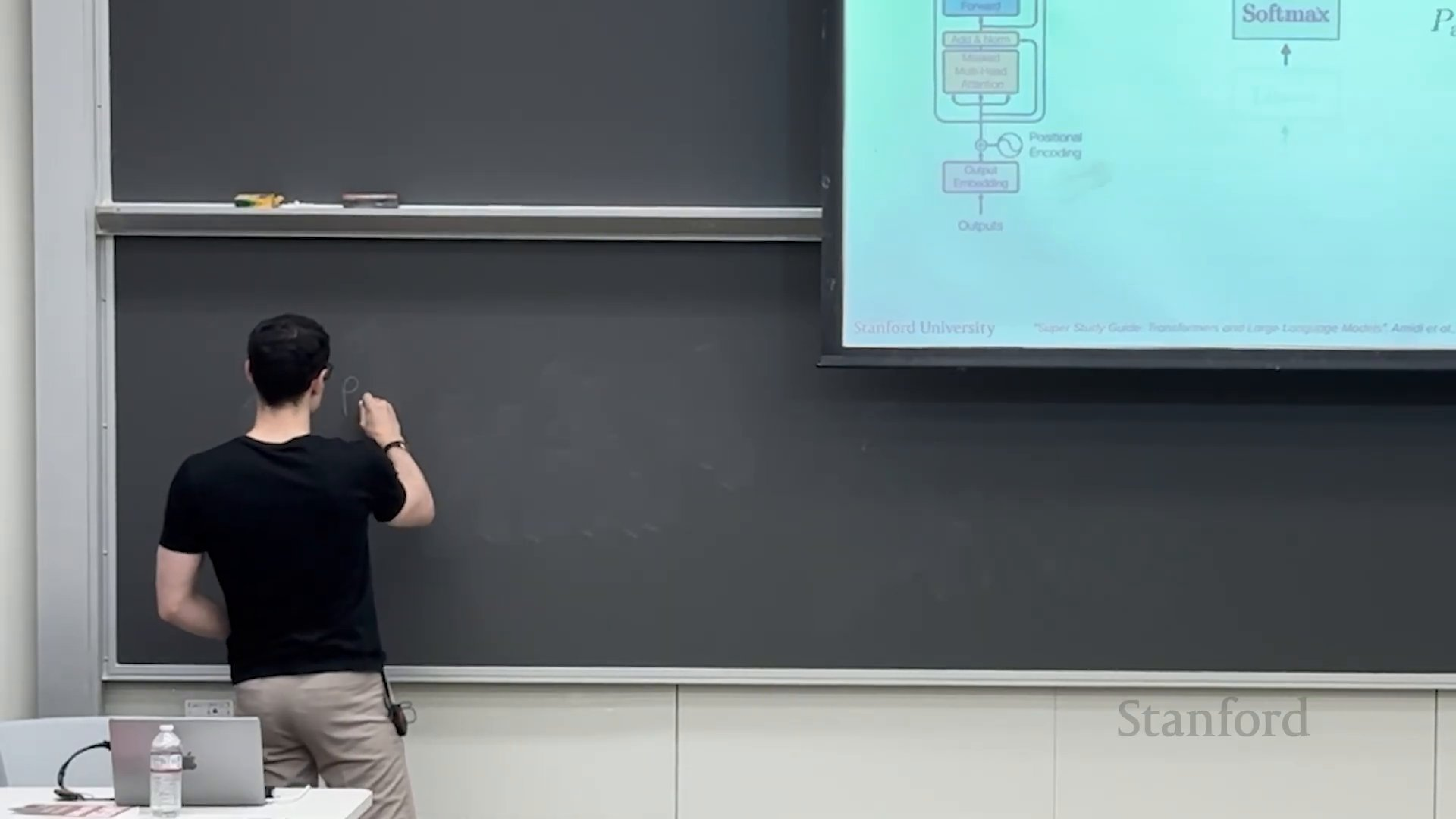
Softmax

$$P_{\text{adj}}(w_{t+1} = w_i | C) = \frac{\exp\left(\frac{x_i}{T}\right)}{\sum_{j=1}^{n} \exp\left(\frac{x_j}{T}\right)}$$

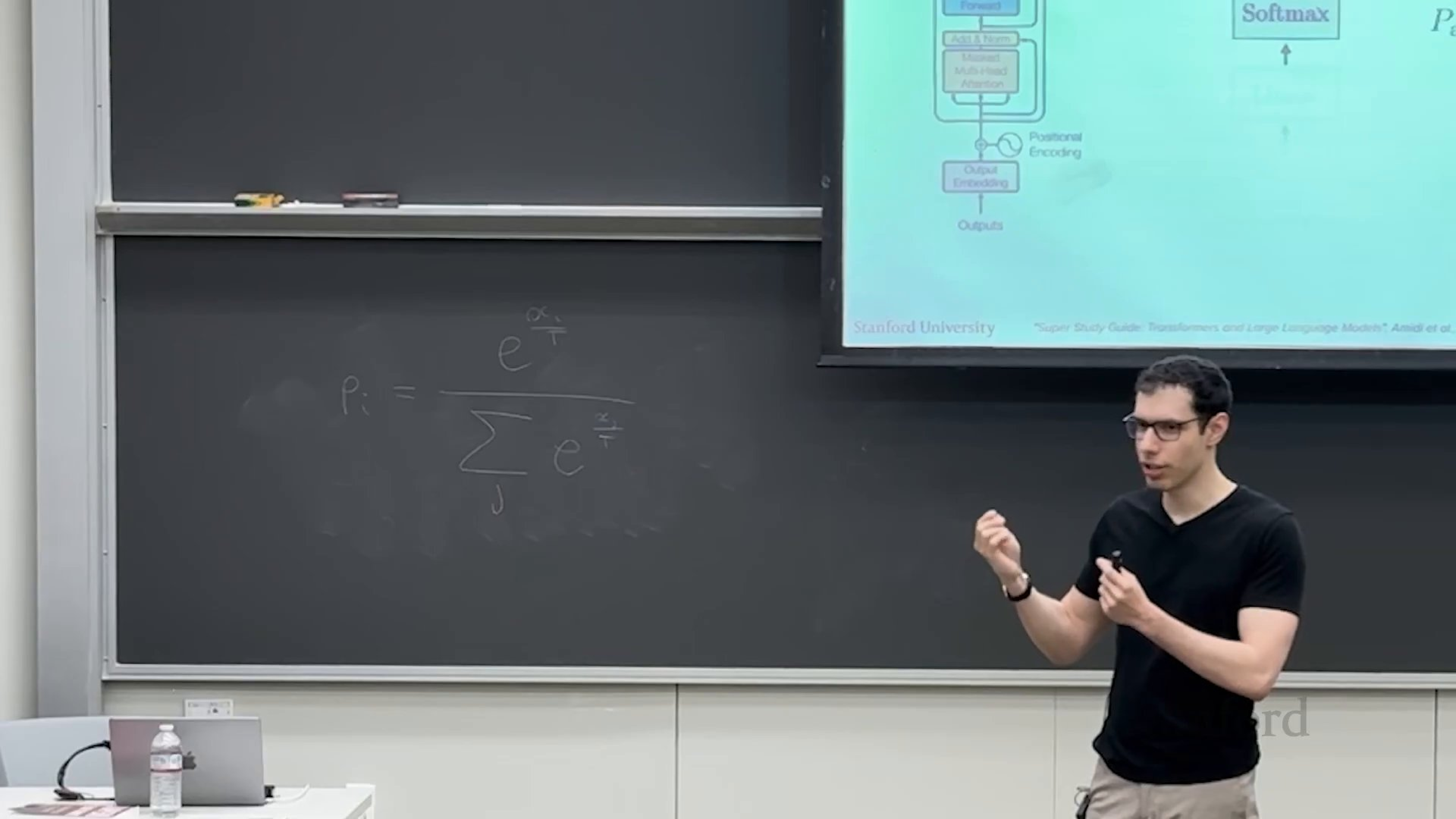

Temperature allows to tweak output probabilities
Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Masked Multi-Head Attention
Positional Encoding
Output Embedding
Outputs
Softmax
$P_{\text{adj}}(w_{t+1} = w_i | C) = \dfrac{\exp\left(\frac{x_i}{T}\right)}{\sum_{j=1}^{n} \exp\left(\frac{x_j}{T}\right)}$
Stanford University
Amidi et al., 2024
Stanford

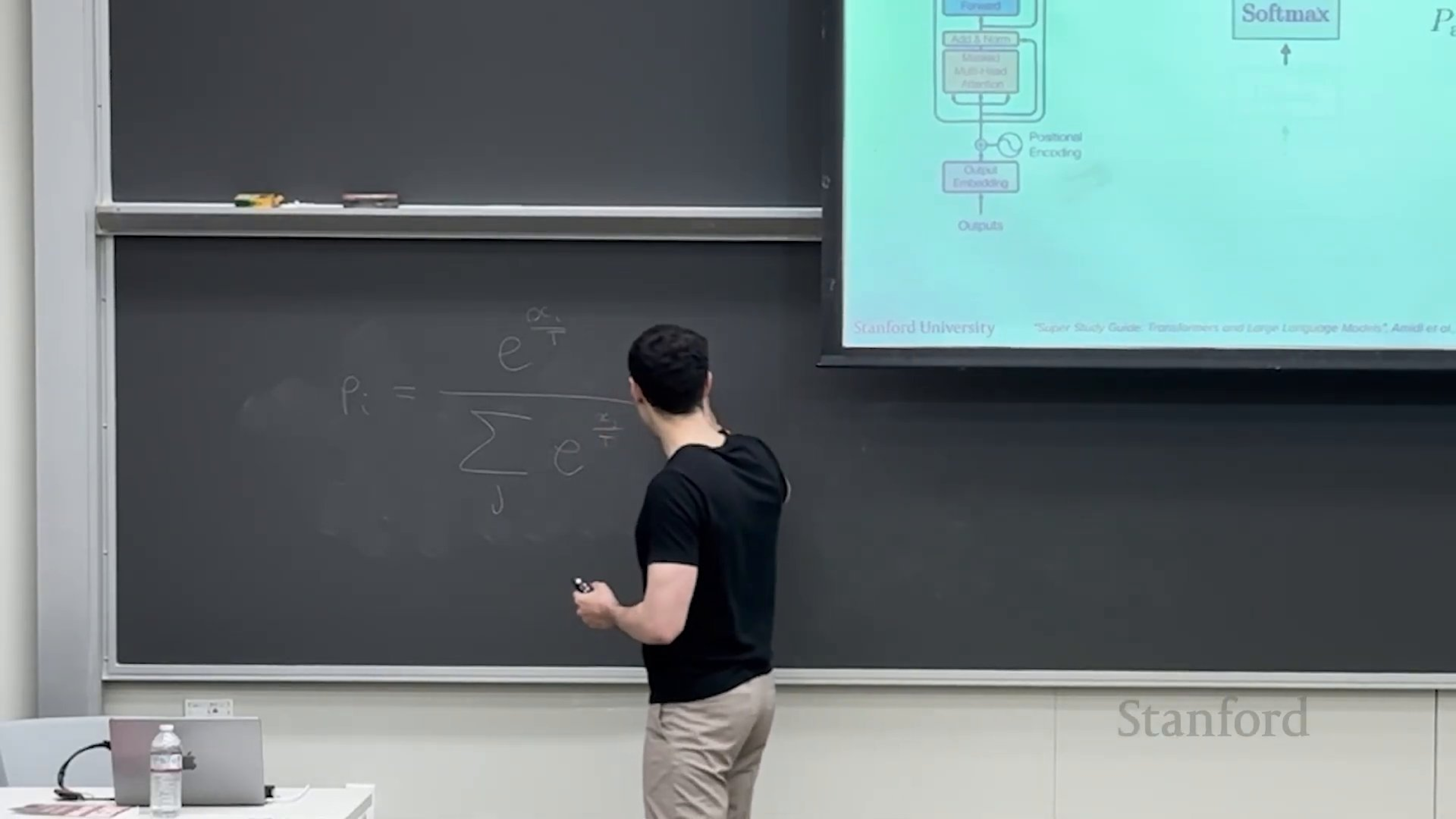

Temperature allows to tweak output probabilities
Output Probabilities
Positional Encoding
Output Embedding
Outputs
Softmax
$$P_{\text{adj}}(w_{t+1} = w_i | C) = \frac{\exp\left(\frac{x_i}{T}\right)}{\sum_{j=1}^{n} \exp\left(\frac{x_j}{T}\right)}$$
Stanford University
"Super Study Guide: Transformers and Large Language Models," Amidi et al., 2024
Stanford

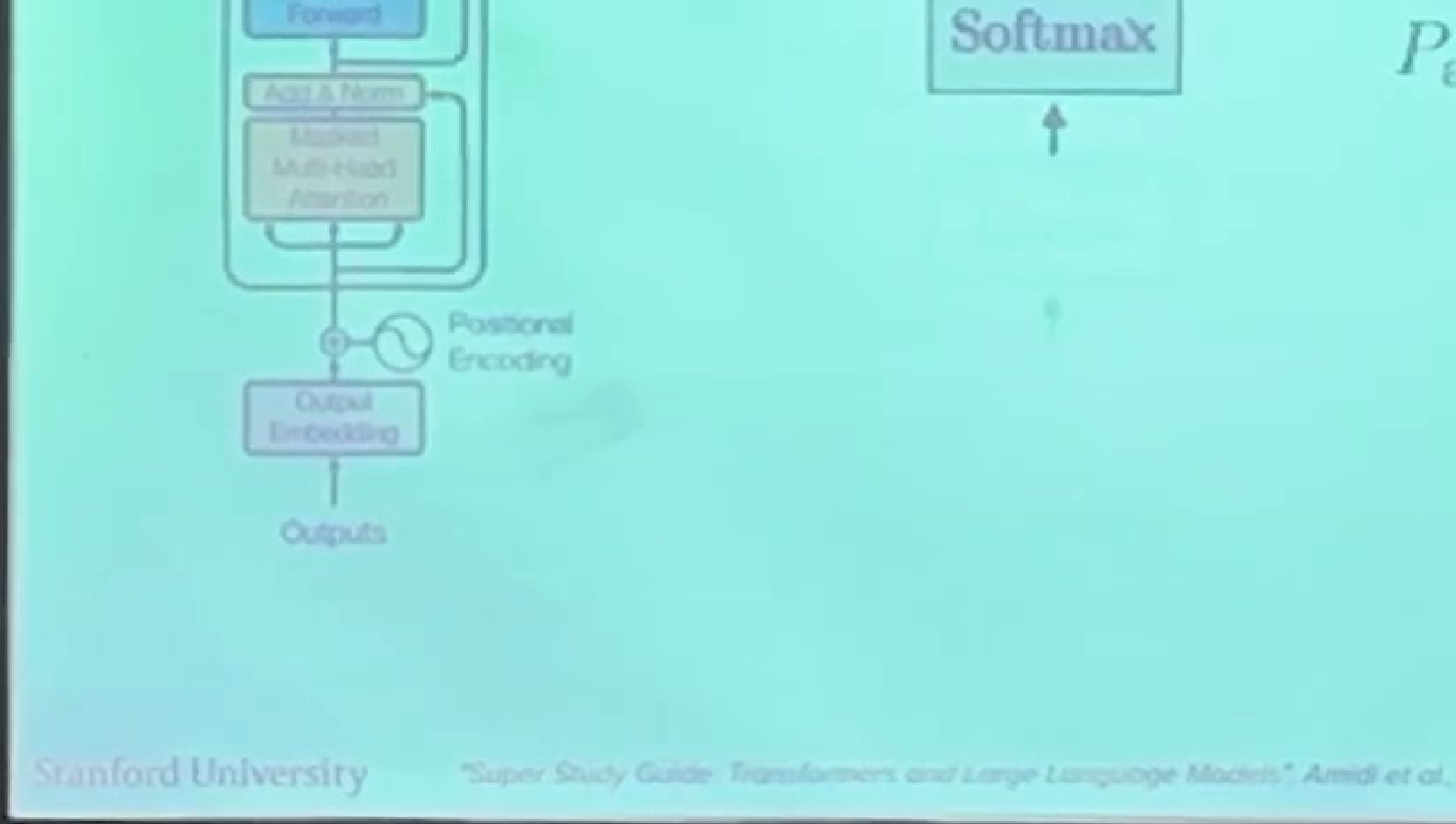

Forward
Add & Norm
Masked
Multi-Head
Attention
Positional
Encoding
Output
Embedding
Outputs
Softmax
Stanford University
"Super Study Guide: Transformers and Large Language Models", Amidi et al.


Forward
Add & Norm
Masked
Multi-Head
Attention
Positional
Encoding
Output
Embedding
Outputs
Softmax
Stanford University
"Super Study Guide: Transformers and Large Language Models", Amidi et al.
Stanford

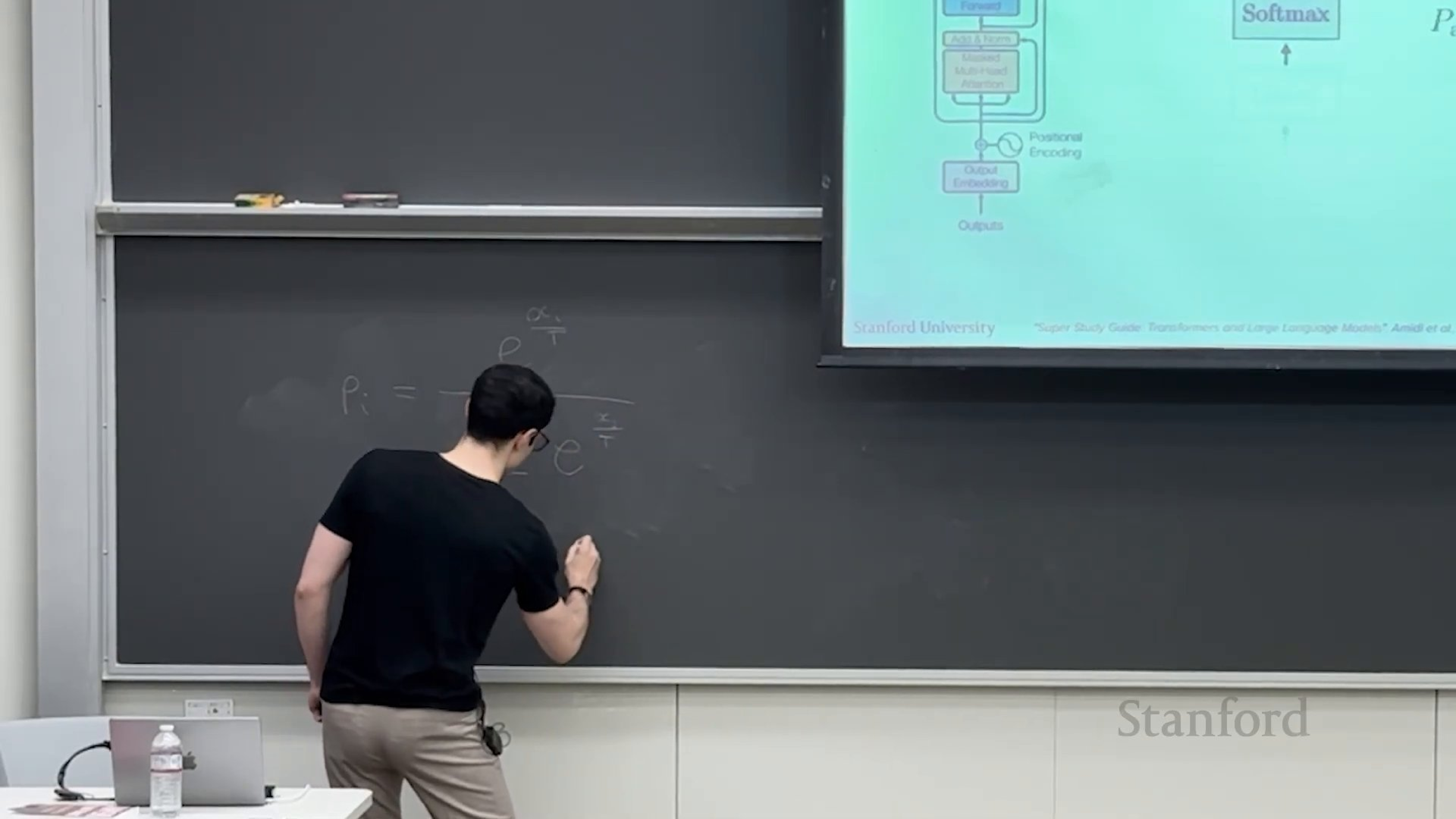

Forward
Add & Norm
Masked Multi-Head Attention
Positional Encoding
Output Embedding
Outputs
Softmax
Stanford University
"Super Study Guide: Transformers and Large Language Models", Amidi et al.
$p_i = \frac{e^{\frac{x_i}{T}}}{\sum_j e^{\frac{x}{T}}}$
Stanford

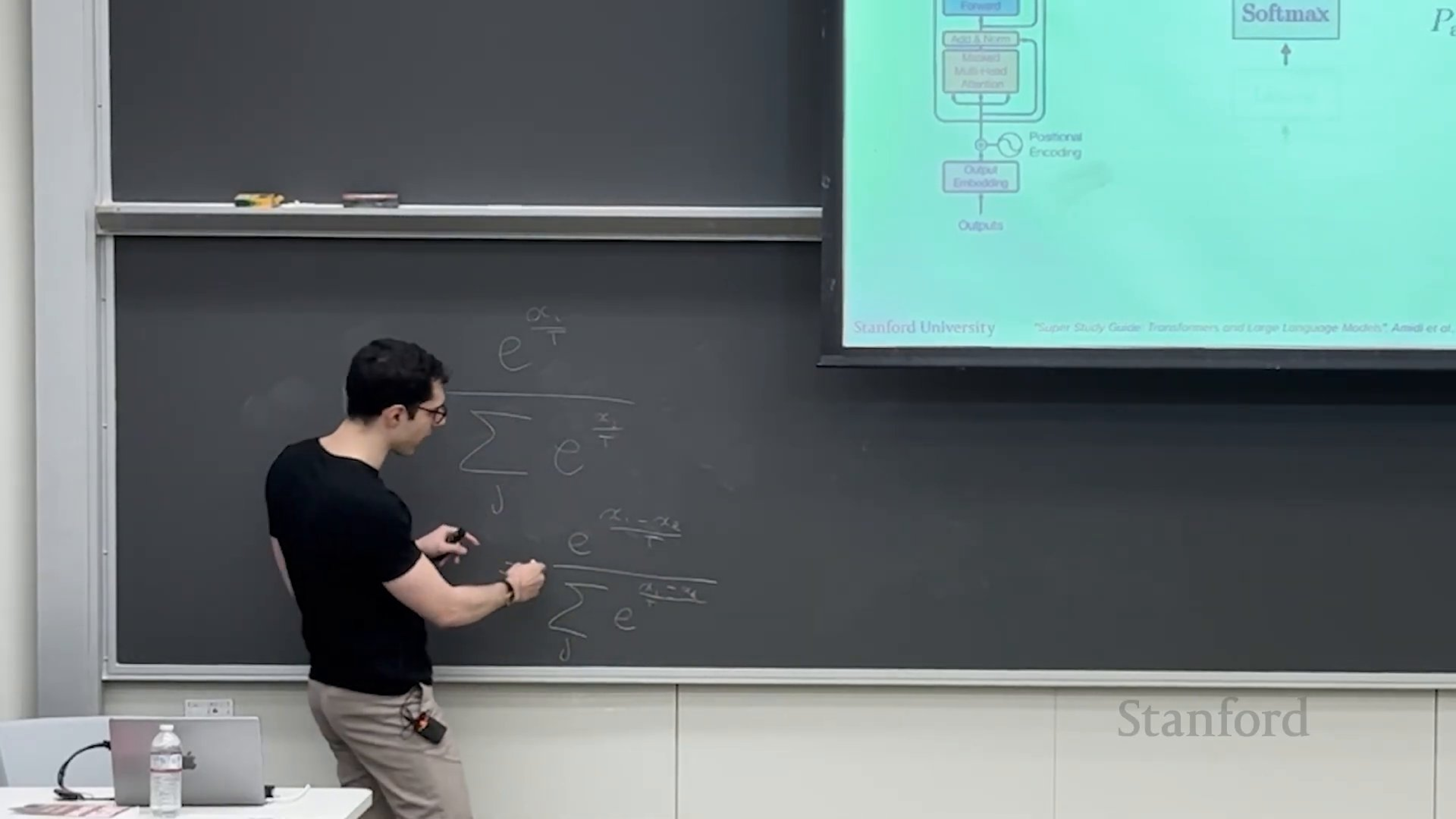

Forward
Add & Norm
Masked Multi-Head Attention
Positional Encoding
Output Embedding
Outputs
Softmax
Stanford University
"Super Study Guide: Transformers and Large Language Models", Amidi et al.
$p_i = \frac{e^{\frac{x_i}{T}}}{\sum_j e^{\frac{x_j}{T}}}$


Forward
Add & Norm
Masked
Multi-Head
Attention
Positional
Encoding
Output
Embedding
Outputs
Softmax
Stanford University
"Super Study Guide: Transformers and Large Language Models", Amidi et al.
$p_i = \frac{e^{\frac{x_i}{T}}}{\sum_j e^{\frac{x_j}{T}}}$
Stanford



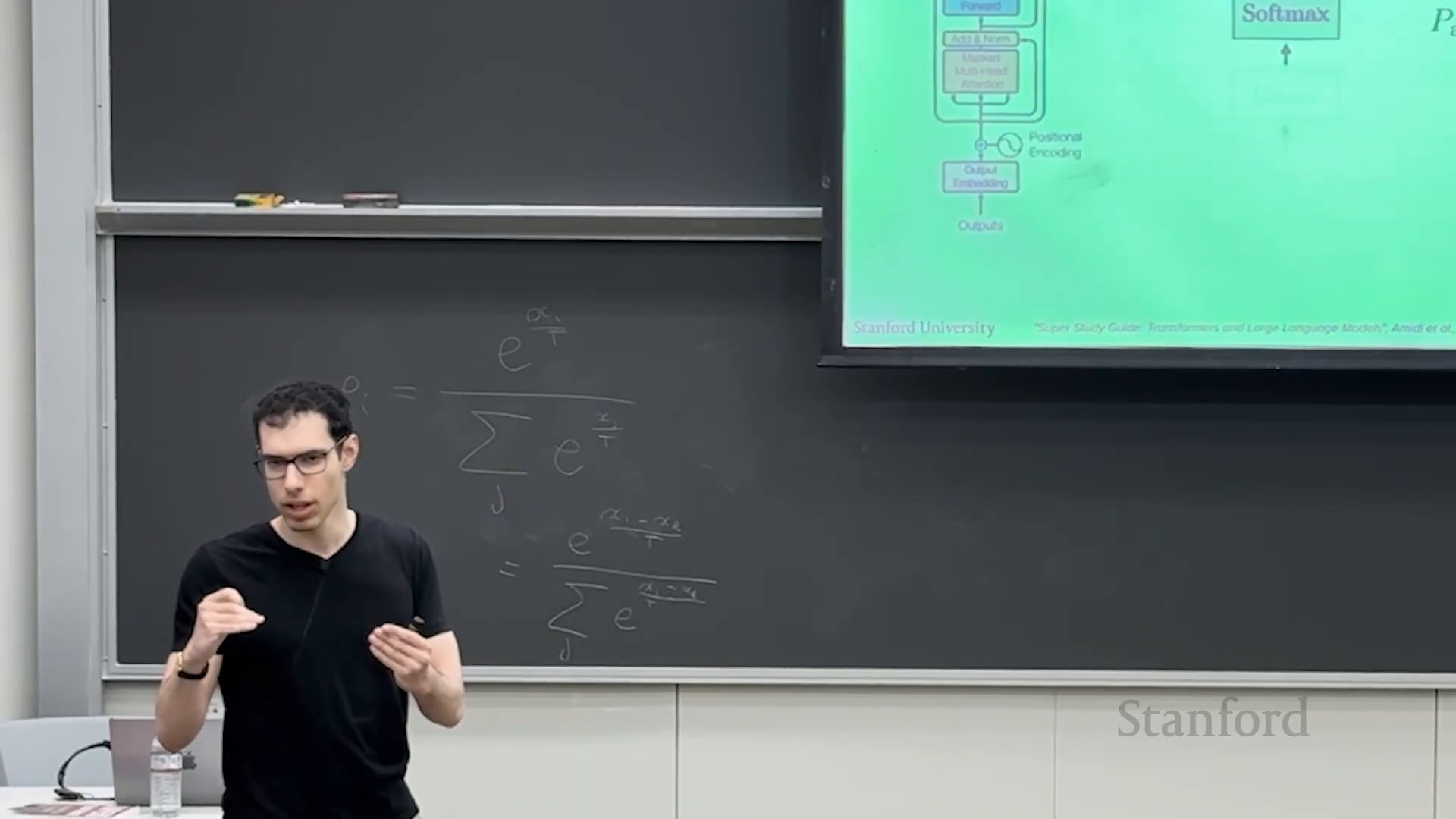
$$p_i = \frac{e^{\frac{x_i}{T}}}{\sum_j e^{\frac{x_j}{T}}} =$$

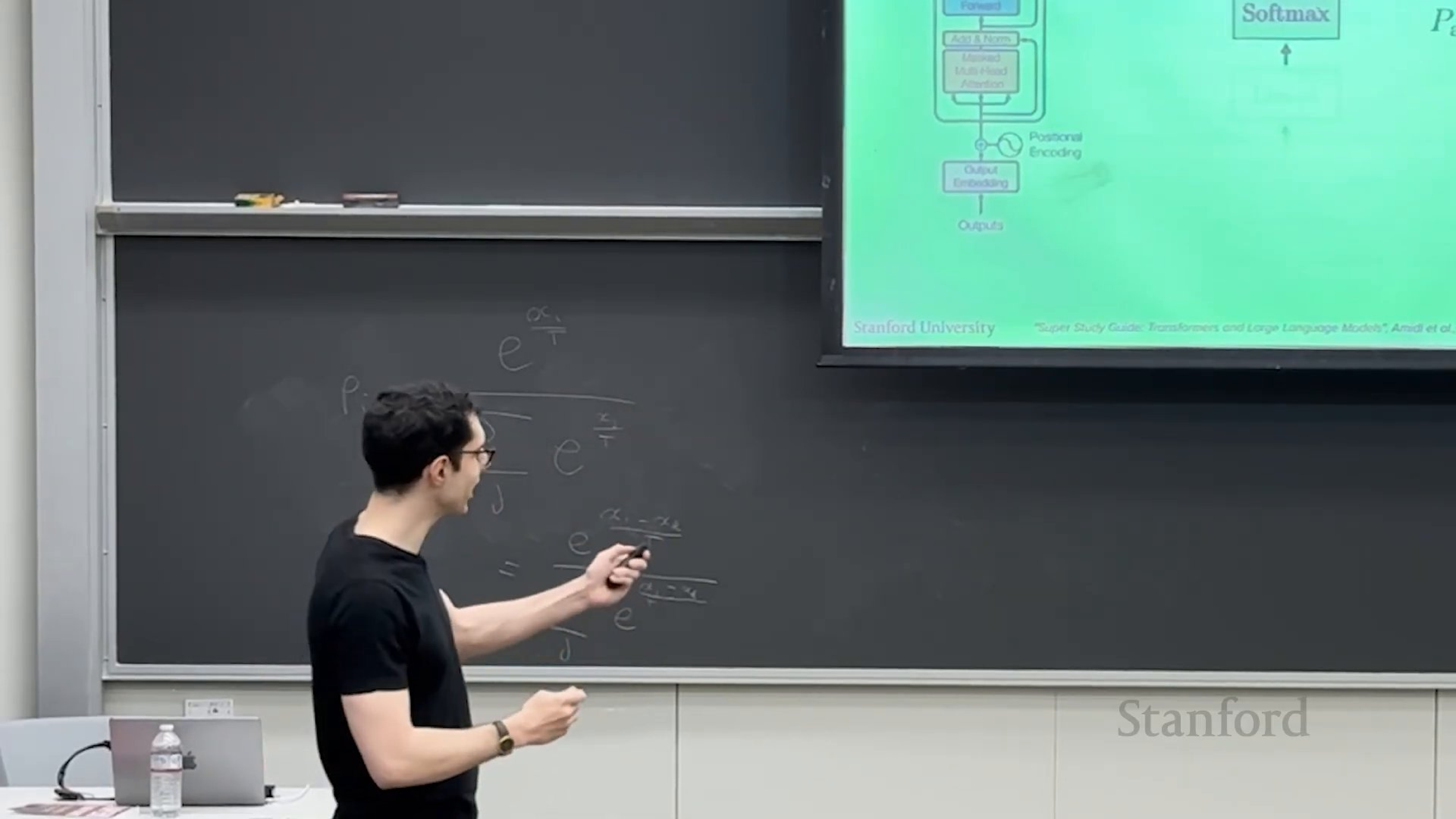

Forward
Add & Norm
Masked
Multi-Head
Attention
Positional
Encoding
Output
Embedding
Outputs
Softmax
Stanford University
"Super Study Guide: Transformers and Large Language Models", Amidi et al.
Stanford



Forward
Add & Norm
Masked Multi-Head Attention
Positional Encoding
Output Embedding
Outputs
Softmax
Stanford University
"Super Study Guide: Transformers and Large Language Models", Amidi et al.
$$\frac{e^{\frac{x_i}{T}}}{\sum_j e^{\frac{x_j}{T}}}$$
$$\frac{e^{\frac{x_i - x_k}{T}}}{\sum_j e^{\frac{x_j - x_k}{T}}}$$
Stanford

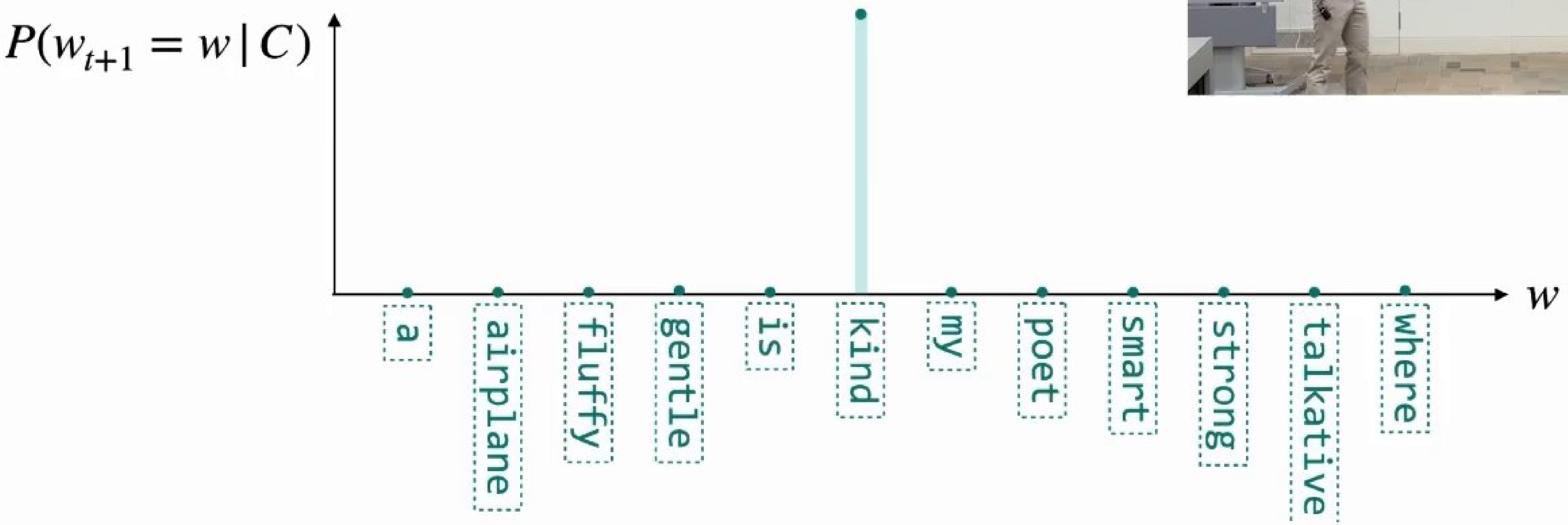

Forward
Add & Norm
Masked
Multi-Head
Attention
Positional
Encoding
Output
Embedding
Outputs
Softmax
Stanford University
"Super Study Guide: Transformers and Large Language Models", Amidi et al.
Stanford


Forward
Add & Norm
Masked Multi-Head Attention
Positional Encoding
Output Embedding
Outputs
Softmax
Stanford University
"Super Study Guide: Transformers and Large Language Models", Amidi et al.
$o_i = \frac{e^{\frac{x_i}{T}}}{\sum_j e^{\frac{x_j}{T}}} = \frac{e^{\frac{x_i - x_k}{T}}}{\sum_j e^{\frac{x_j - x_k}{T}}}$
Stanford

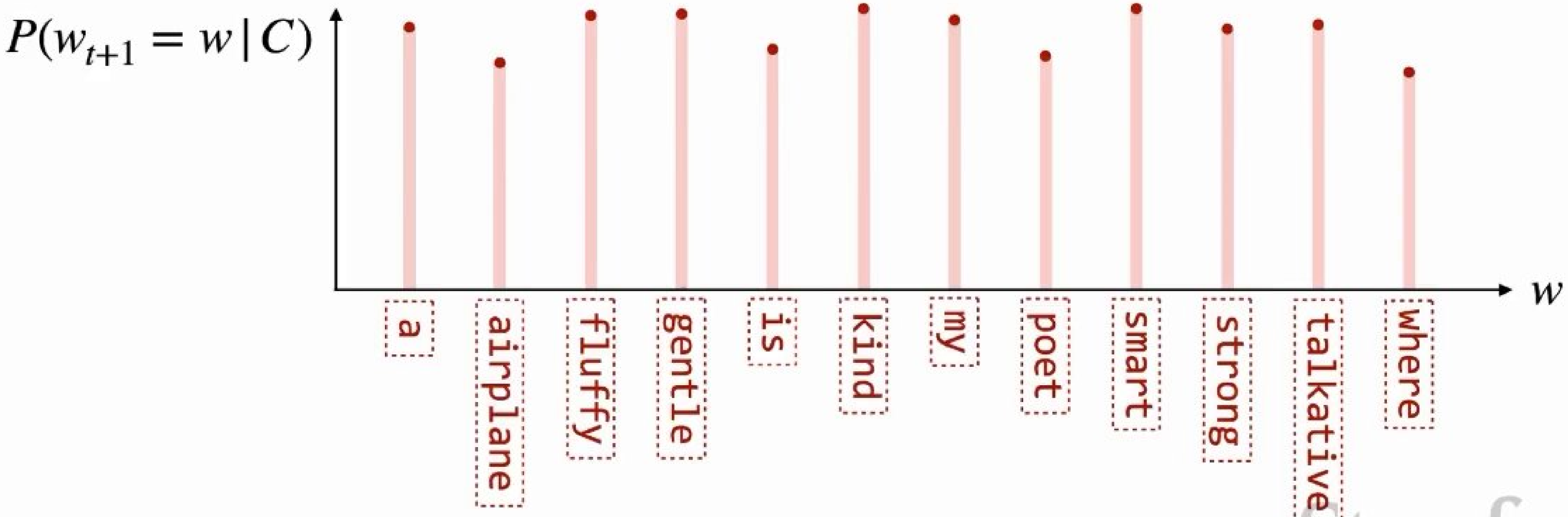

Forward
Add & Norm
Masked Multi-Head Attention
Positional Encoding
Output Embedding
Outputs
Softmax
Stanford University
"Super Study Guide: Transformers and Large Language Models", Amidi et al.
Stanford

Impact of temperature on probabilities

Small T

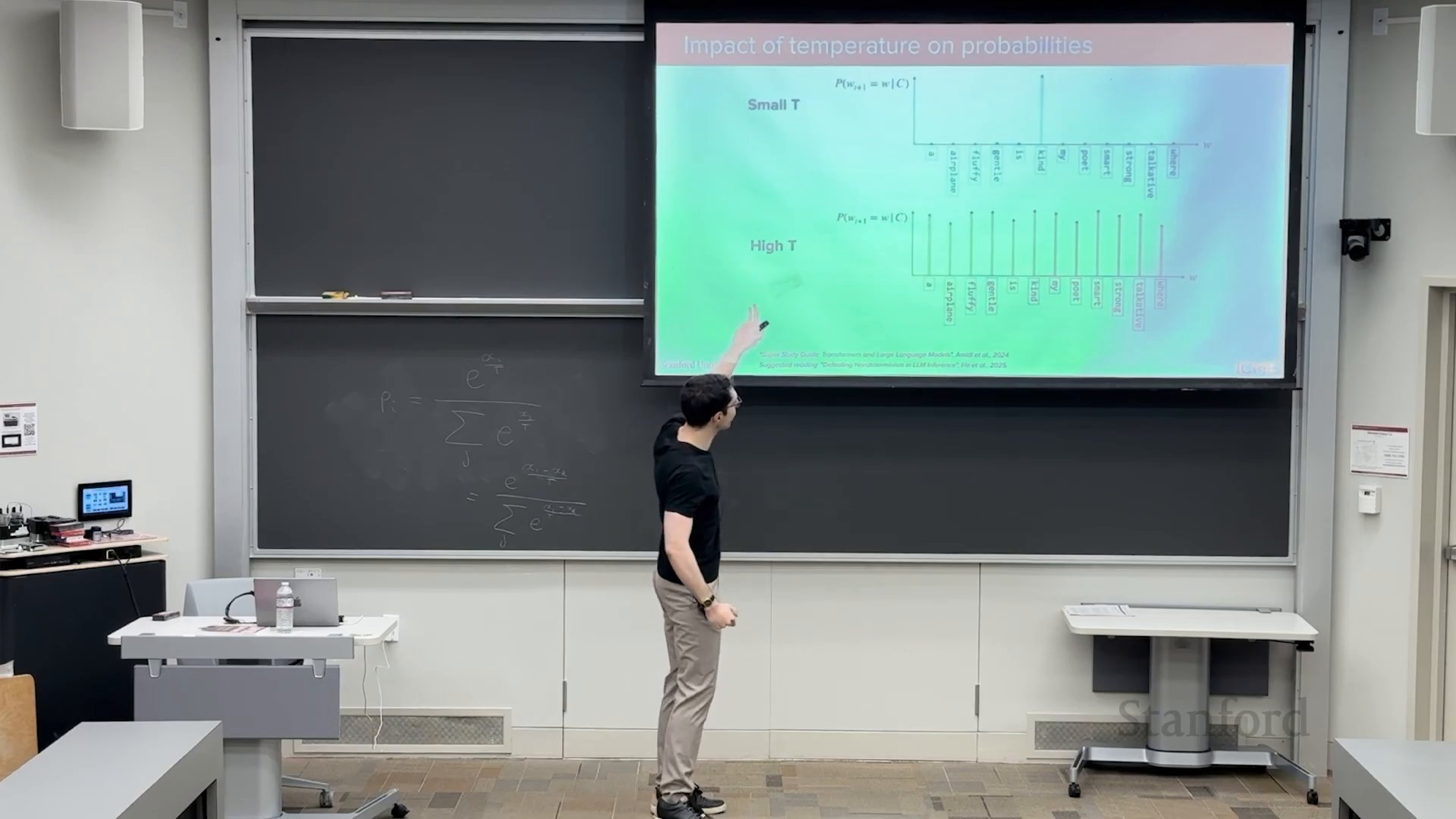


High T

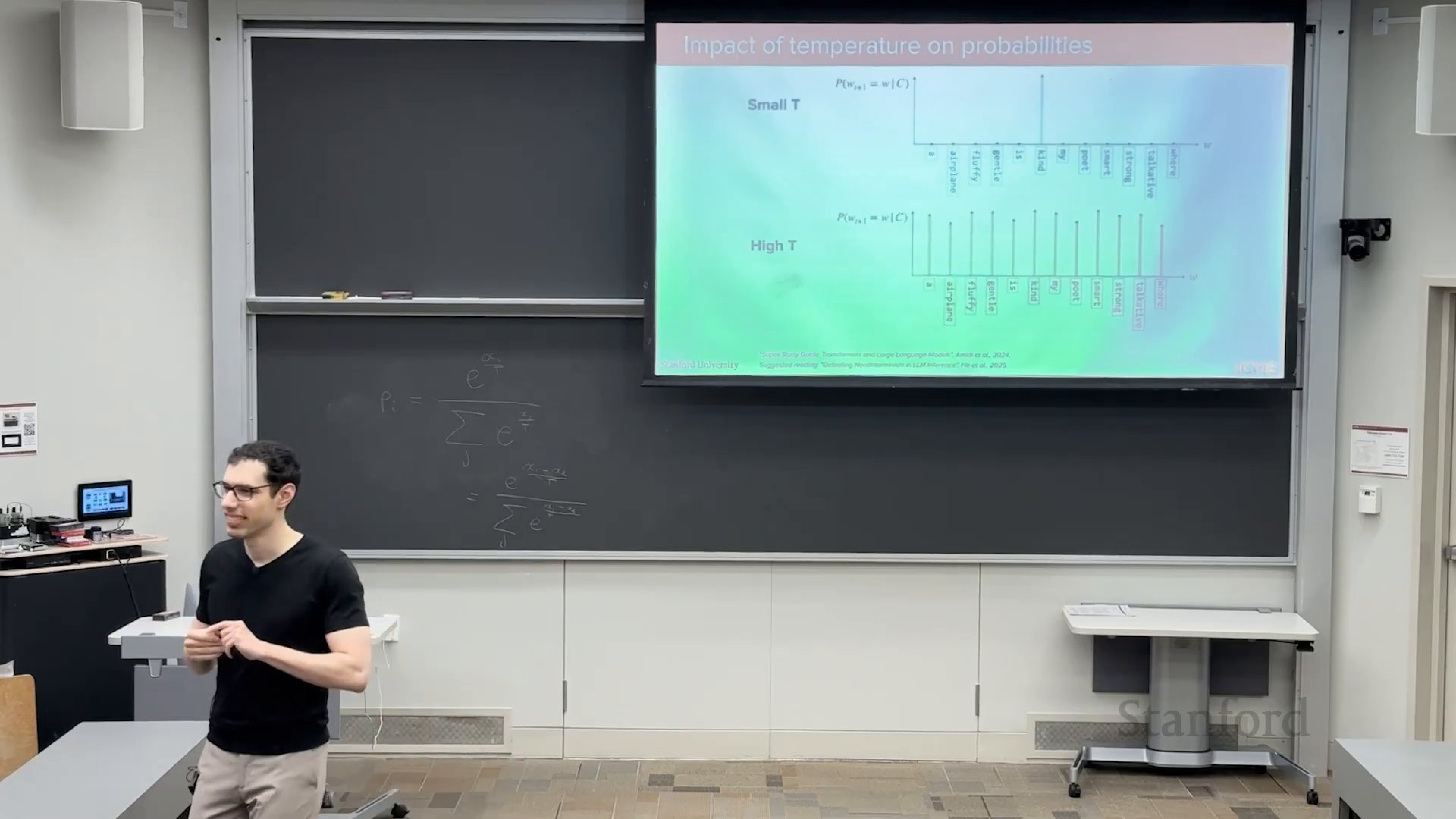


"Super Study Guide: Transformers and Large Language Models", Amidi et al., 2024.

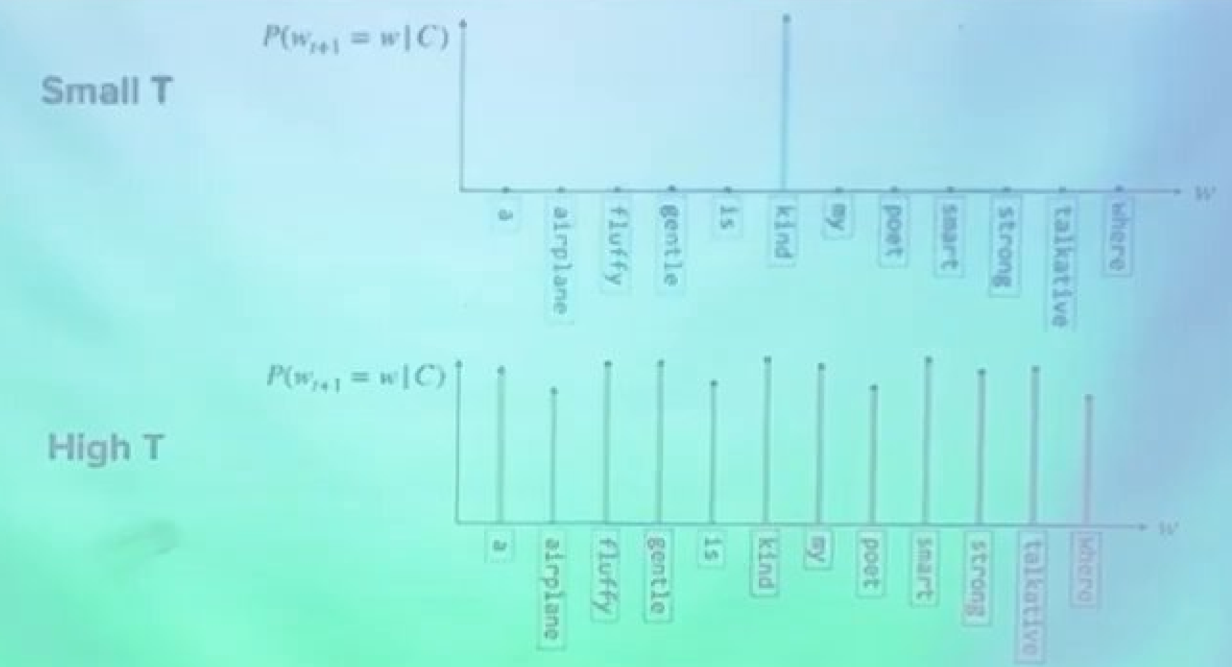
Suggested reading: "Defeating Nondeterminism in LLM Inference", He et al., 2025.



Impact of temperature on probabilities
Small T
$P(w_{i+1} = w \mid C)$
w
a
airplane
fluffy
gentle
is
kind
my
poet
smart
strong
talkative
where
High T
$P(w_{i+1} = w \mid C)$
w
a
airplane
fluffy
gentle
is
kind
my
poet
smart
strong
talkative
where
"Super Study Guide: Transformers and Large Language Models", Amidi et al., 2024.
Suggested reading: "Defeating Nondeterminism in LLM Inference", He et al., 2025.
$p_i = \frac{e^{\frac{x_i}{T}}}{\sum_j e^{\frac{x_j}{T}}} = \frac{e^{\frac{x_i - x_k}{T}}}{\sum_j e^{\frac{x_j - x_k}{T}}}$
Stanford


Impact of temperature on probabilities
Small T
$P(w_{t+1} = w \mid C)$
a
airplane
fluffy
gentle
is
kind
my
poet
smart
strong
talkative
where
High T
$P(w_{t+1} = w \mid C)$
a
airplane
fluffy
gentle
is
kind
my
poet
smart
strong
talkative
where
"Super Study Guide: Transformers and Large Language Models", Amidi et al., 2024.
Suggested reading: "Defeating Nondeterminism in LLM Inference", He et al., 2025.
Stanford University
ICME
$p_i = \frac{e^{\frac{x_i}{T}}}{\sum_j e^{\frac{x_j}{T}}} = \frac{e^{\frac{x_i - x_k}{T}}}{\sum_j e^{\frac{x_j - x_k}{T}}}$
Stanford

Impact of temperature on probabilities

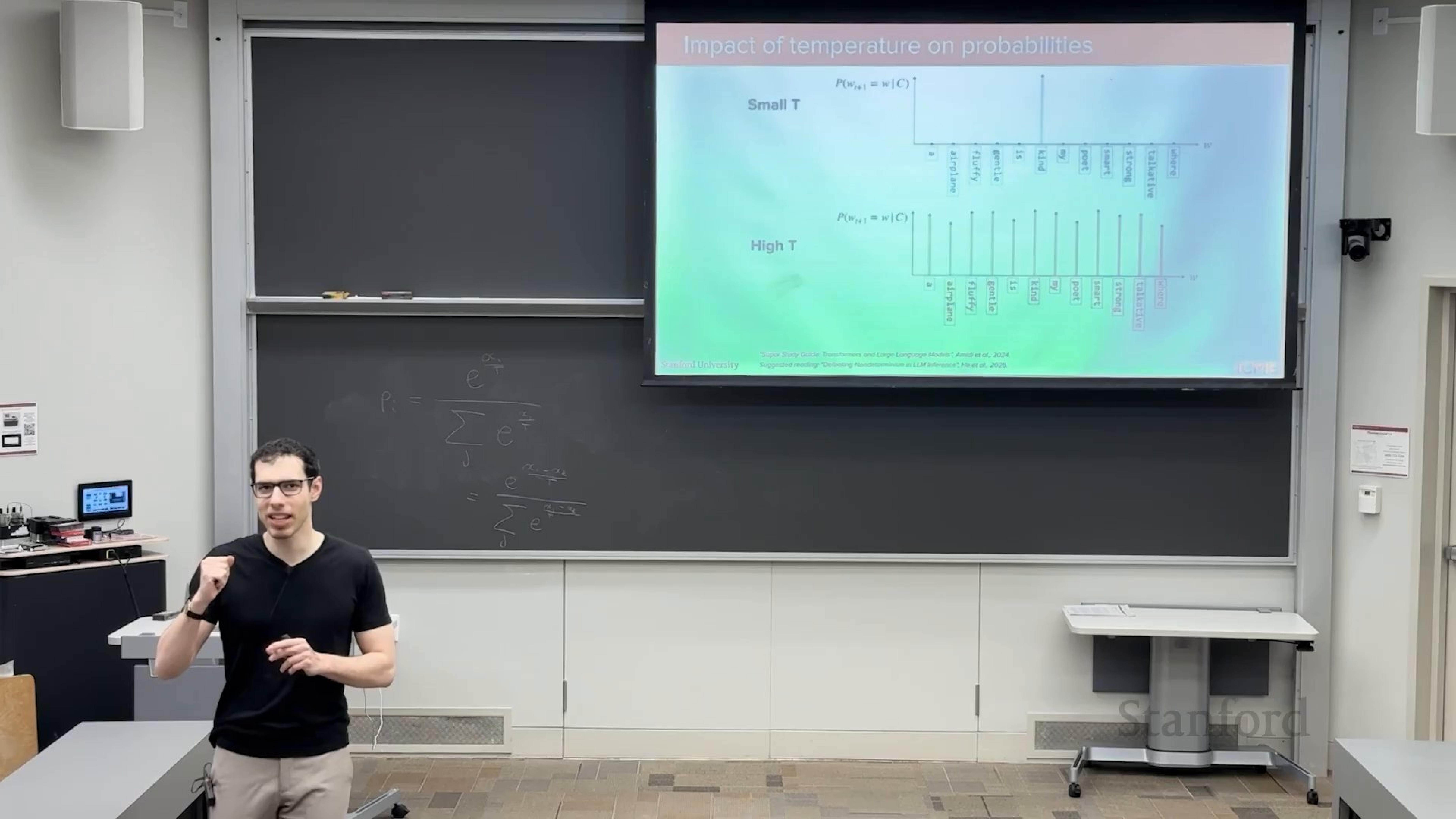


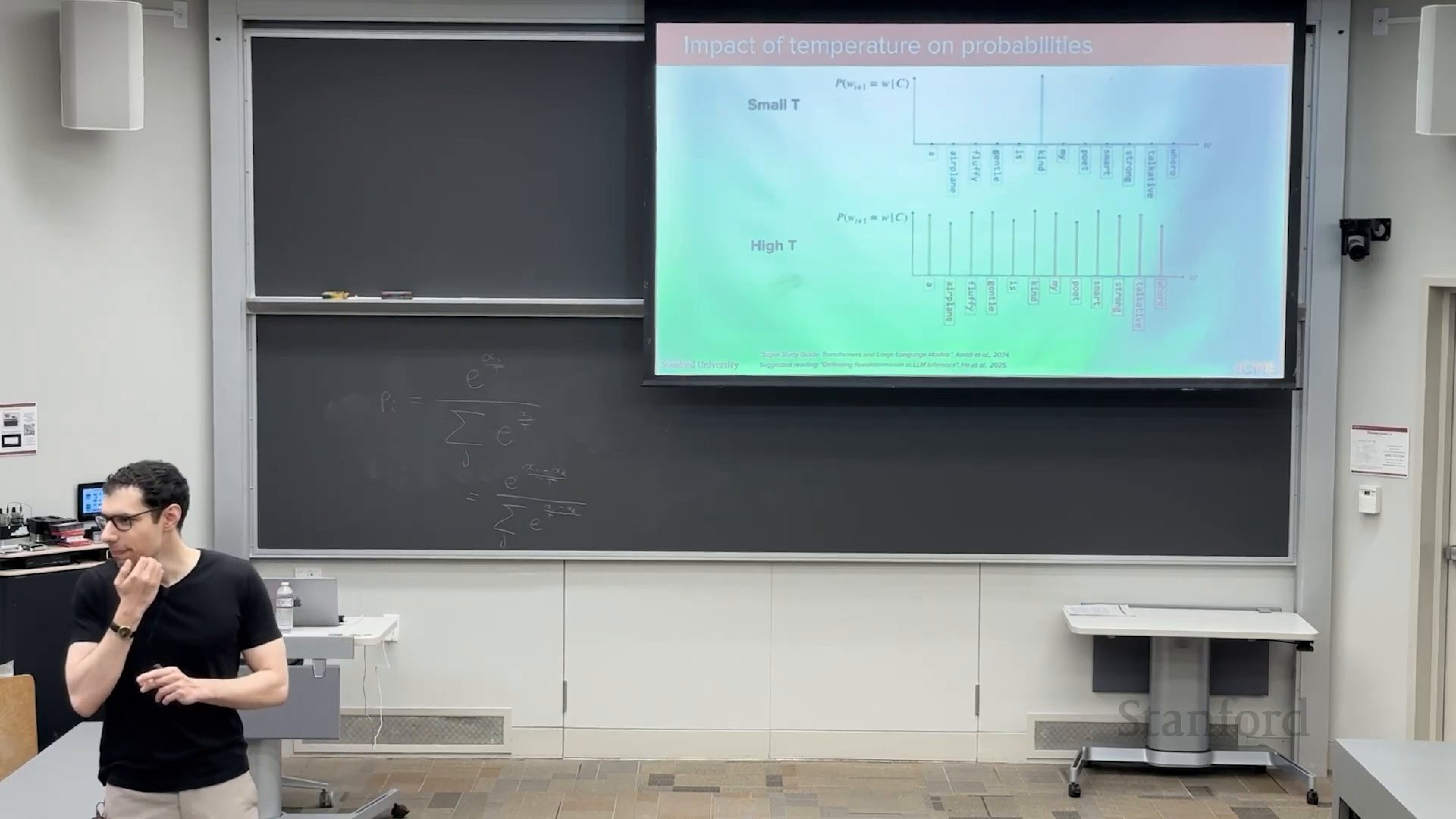
"Super Study Guide: Transformers and Large Language Models", Amidi et al., 2024.
Suggested reading: "Defeating Nondeterminism in LLM Inference", He et al., 2025.

Stanford University

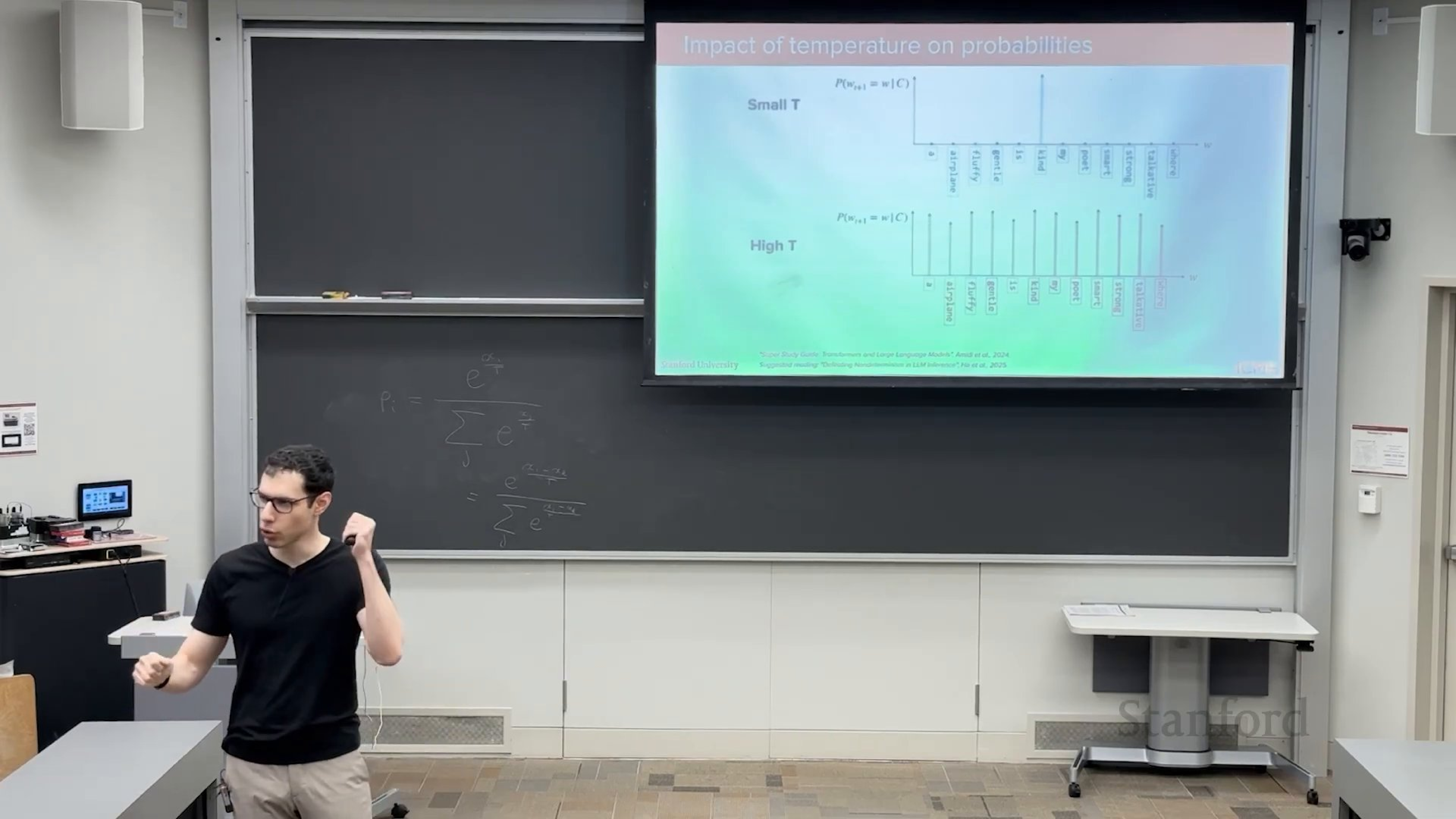
ICME

$$p_i = \frac{e^{\frac{x_i}{T}}}{\sum_j e^{\frac{x_j}{T}}} = \frac{e^{\frac{x_i - x_k}{T}}}{\sum_j e^{\frac{x_j - x_k}{T}}}$$


Impact of temperature on probabilities
Small T
$P(w_{t+1} = w \mid C)$
a
airplane
fluffy
gentle
is
kind
my
poet
smart
strong
talkative
where
w
High T
$P(w_{t+1} = w \mid C)$
a
airplane
fluffy
gentle
is
kind
my
poet
smart
strong
talkative
where
w
"Super Study Guide: Transformers and Large Language Models", Amidi et al., 2024.
Suggested reading: "Defeating Nondeterminism in LLM Inference", He et al., 2025.
Stanford University
ICME
$p_i = \frac{e^{\frac{x_i}{T}}}{\sum_j e^{\frac{x_j}{T}}} = \frac{e^{\frac{x_i - x_k}{T}}}{\sum_j e^{\frac{x_j - x_k}{T}}}$
Stanford


Impact of temperature on probabilities
Small T
$P(w_{t+1} = w \mid C)$
w
a
airplane
fluffy
gentle
is
kind
my
poet
smart
strong
talkative
where
High T
$P(w_{t+1} = w \mid C)$
w
a
airplane
fluffy
gentle
is
kind
my
poet
smart
strong
talkative
where
"Super Study Guide: Transformers and Large Language Models", Amidi et al., 2024.
Suggested reading: "Defeating Nondeterminism in LLM Inference", He et al., 2025.
Stanford University
$p_i = \frac{e^{\frac{x_i}{T}}}{\sum_j e^{\frac{x_j}{T}}} = \frac{e^{\frac{x_i - x_k}{T}}}{\sum_j e^{\frac{x_j - x_k}{T}}}$
Stanford


Impact of temperature on probabilities
Small T
$P(w_{t+1} = w \mid C)$
w
a
airplane
fluffy
gentle
is
kind
my
poet
smart
strong
talkative
where
High T
$P(w_{t+1} = w \mid C)$
w
a
airplane
fluffy
gentle
is
kind
my
poet
smart
strong
talkative
where
"Super Study Guide: Transformers and Large Language Models", Amidi et al., 2024
Suggested reading: "Defeating Nondeterminism in LLM Inference", He et al., 2025
Stanford University
Stanford



Impact of temperature on probabilities
Small T
$P(w_{t+1} = w \mid C)$
a
airplane
fluffy
gentle
is
kind
my
poet
smart
strong
talkative
where
w
High T
$P(w_{t+1} = w \mid C)$
a
airplane
fluffy
gentle
is
kind
my
poet
smart
strong
talkative
where
w
"Super Study Guide: Transformers and Large Language Models", Amidi et al., 2024
Suggested reading: "Defeating Nondeterminism in LLM Inference", He et al., 2025
Stanford University
$p_i = \frac{e^{\frac{x_i}{T}}}{\sum_j e^{\frac{x_j}{T}}}$
$= \frac{e^{\frac{x_i - x_k}{T}}}{\sum_j e^{\frac{x_j - x_k}{T}}}$
Stanford


Impact of temperature on probabilities
Small T
$P(w_{t+1} = w \mid C)$
a
airplane
fluffy
gentle
is
kind
my
poet
smart
strong
talkative
where
w
High T
$P(w_{t+1} = w \mid C)$
a
airplane
fluffy
gentle
is
kind
my
poet
smart
strong
talkative
where
w
"Super Study Guide: Transformers and Large Language Models", Amidi et al., 2024
Suggested reading: "Defeating Nondeterminism in LLM Inference", He et al., 2025
Stanford University
ICME
$p_i = \frac{e^{\frac{x_i}{T}}}{\sum_j e^{\frac{x_j}{T}}}$
Stanford

Constraining the output via guided decod

Motivation. Generate output in a specific format

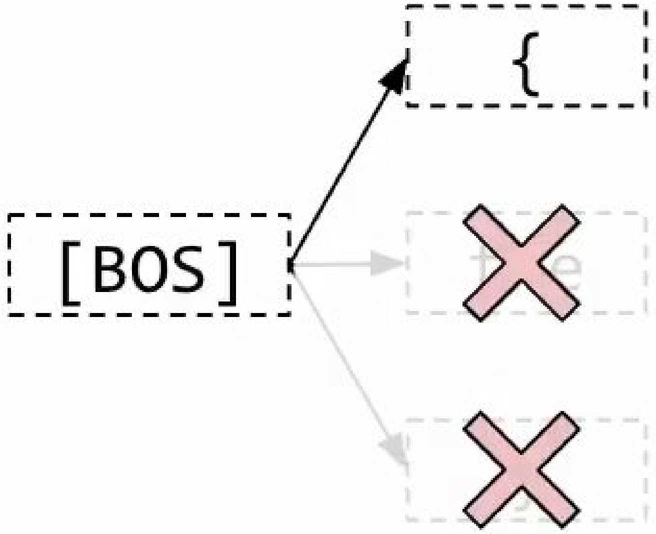
Input prompt
Generate a description of my 33-year old teddy bear who likes reading. *Do this in JSON format.*

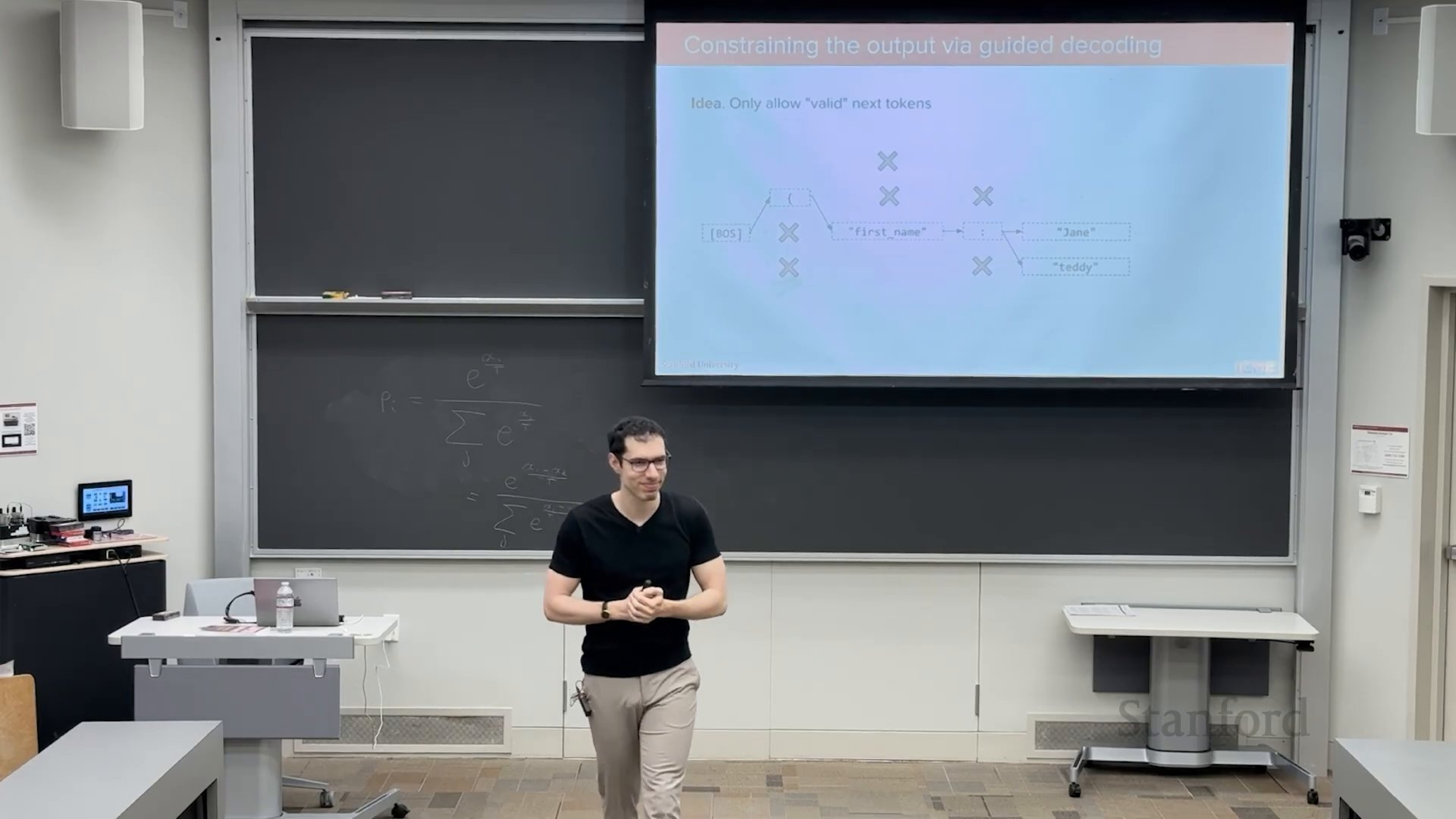
Desired output (JSON)

```json
{
    "first_name": "teddy",
    "last_name": "bear",
    "age": 33,
    "hobby": "reading"
}
```

Constraining the output via guided decod

Idea. Only allow "valid" next tokens




Constraining the output via guided decoding
Idea. Only allow "valid" next tokens
[BOS]
{
"first_name"
:
"Jane"
"teddy"
Stanford University
Stanford


Transformers &
Large Language
Models
Stanford University
ICME
LLM overview
MoE-based LLMs
Response generation
Prompting strategies
Inference optimizations
Stanford

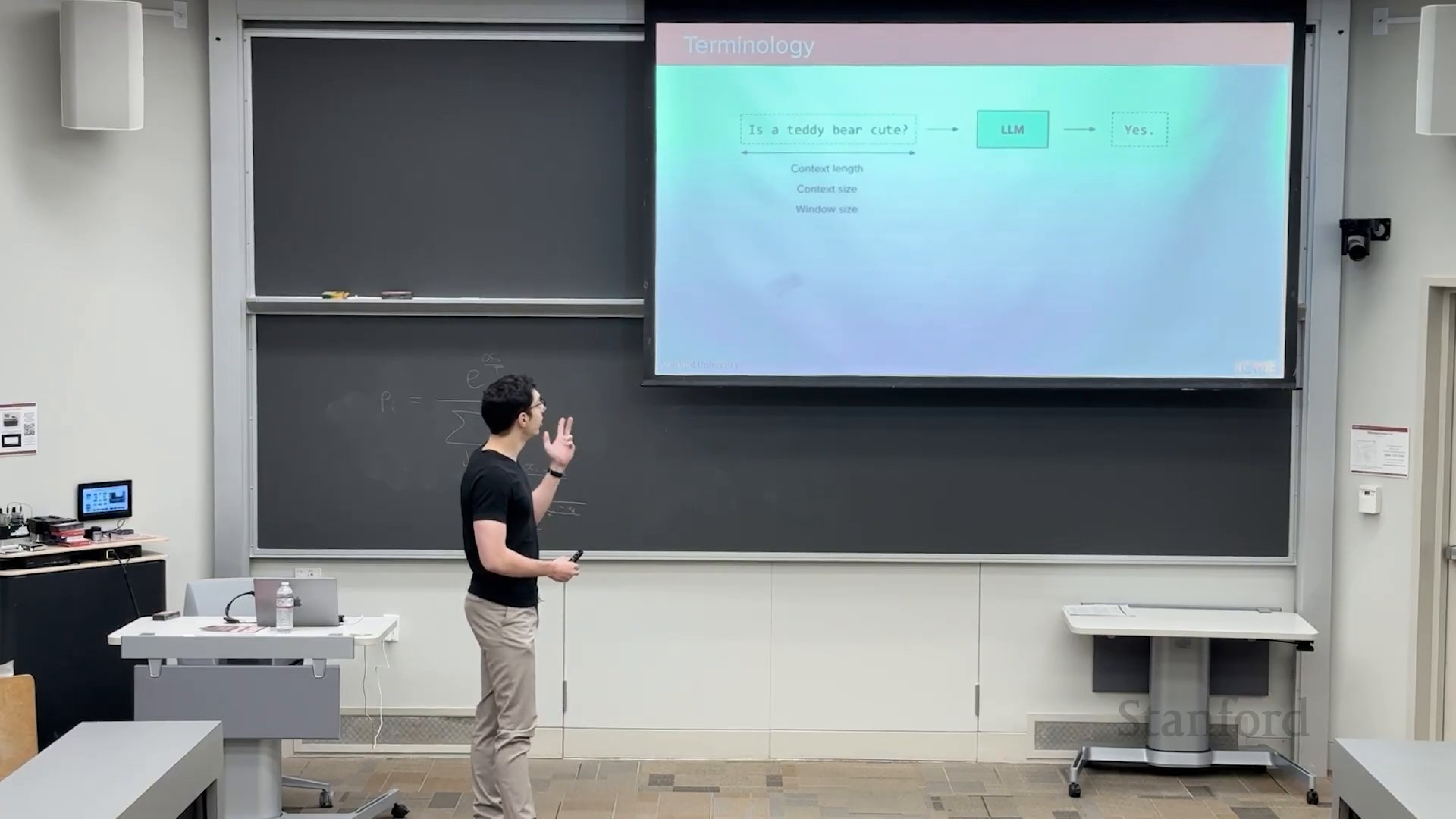
Terminology

`Is a teddy bear cute?` → **LLM** → `Yes.`

Context length

Context size

Window size


Terminology
Is a teddy bear cute?
LLM
Yes.
Context length
Context size
Window size
Stanford University
ICME
Stanford

Terminology

`Is a teddy bear cute?` → **LLM** → `Yes.`

Context length

Context size

Window size

Discussion. Orders of magnitude by:

- Input type
- Models

Stanford


Terminology
Discussion. Orders of magnitude by:
Input type
Models
Beware of "context rot"!
Stanford University
"Context Rot: How Increasing Input Tokens Impacts LLM Performance", Hong et al., 2025
Stanford

Main structure

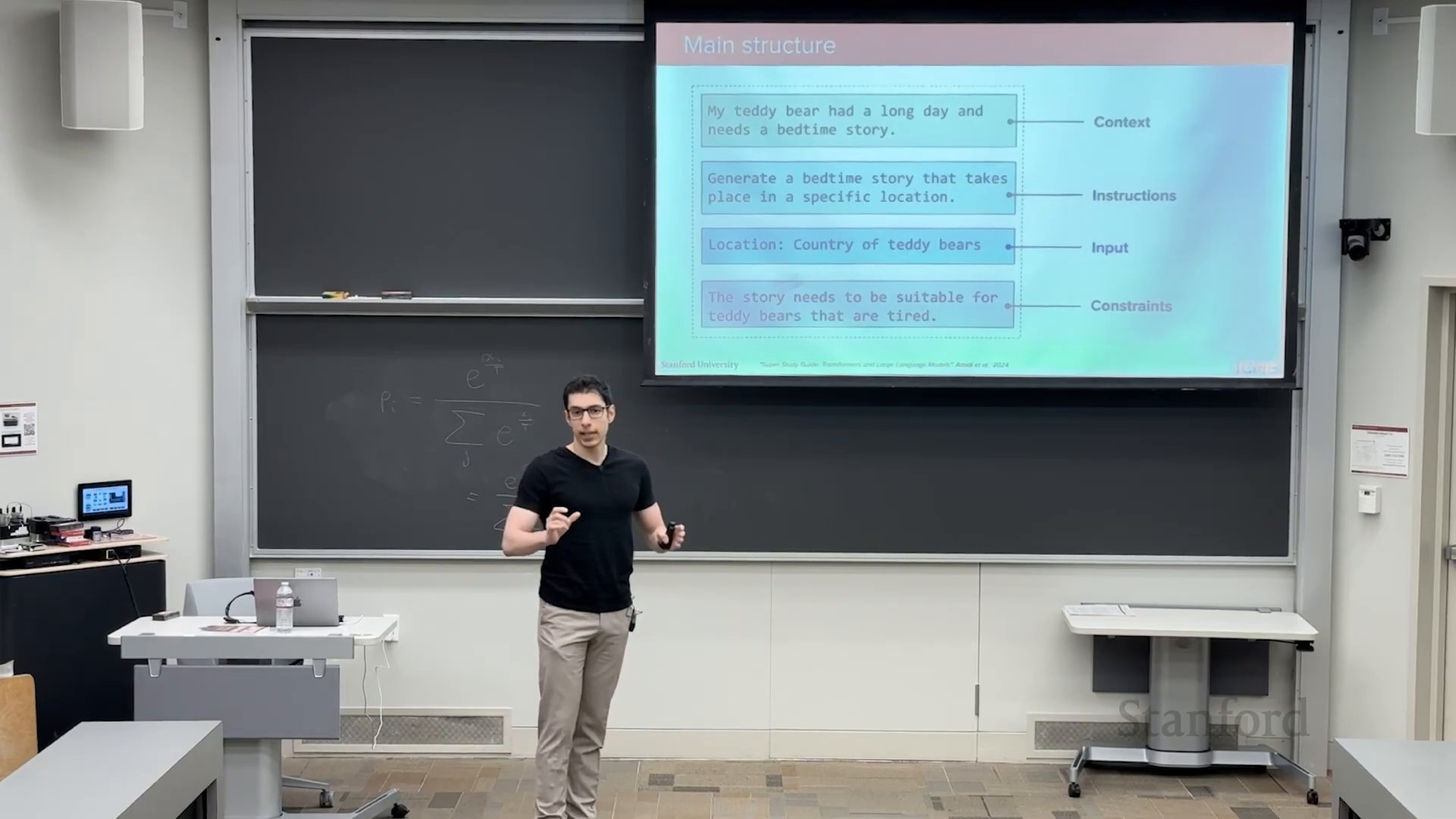
My teddy bear had a long day and needs a bedtime story. — **Context**

Generate a bedtime story that takes place in a specific location. — **Instructions**

Location: Country of teddy bears — **Input**

The story needs to be suitable for teddy bears that are tired. — **Constraints**

Stanford


Main structure
My teddy bear had a long day and needs a bedtime story.
Context
Generate a bedtime story that takes place in a specific location.
Instructions
Location: Country of teddy bears
Input
The story needs to be suitable for teddy bears that are tired.
Constraints
Stanford University
Stanford

In-context learning

ICL = **I**n-**C**ontext **L**earning

Stanford

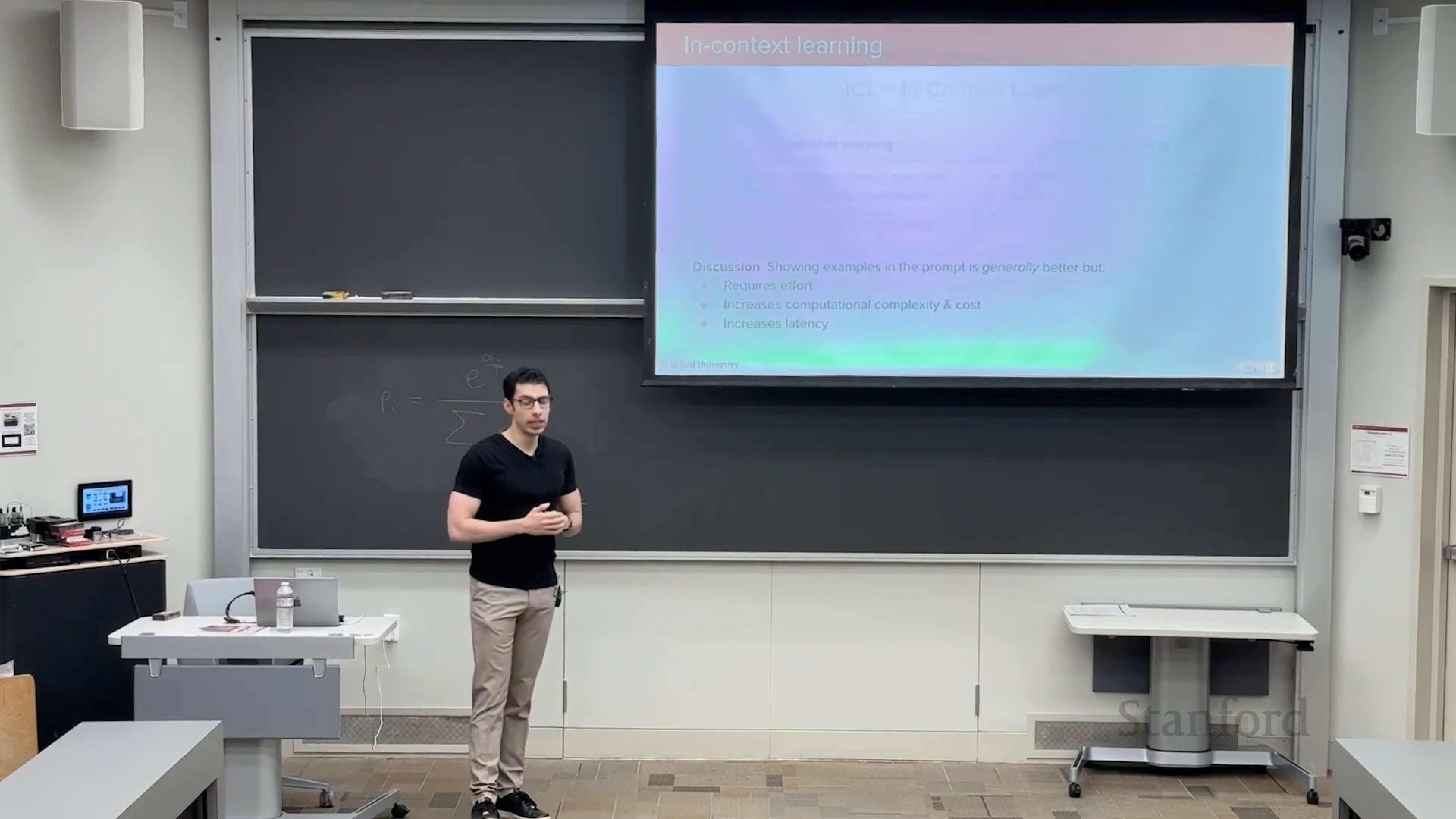

In-context learning
ICL = In-Context Learning
Stanford University
"Language Models are Few-Shot Learners" Brown et al., 2020
Stanford

In-context learning

ICL = In-Context Learning

Zero-shot learning	Few-shot learning
• Question is asked without examples • Performance heavily depends on performance of initial model	• Prompt contains examples of input / output • Typically has a better performance

Stanford


In-context learning
Discussion. Showing examples in the prompt is *generally* better but:
Requires effort
Increases computational complexity & cost
Increases latency
Stanford University
Stanford

Chain of thought

Idea. Explaining reasoning helps in improving performance.

```
Q: How old is this bear?
A: The bear was born in
2020. It is therefore 4.


Q: How old will the bear
be next year?
```

→ LLM →

```
A: It will be one
year older than
its age this year,
which was 4.
Hence, it will be
5.
```

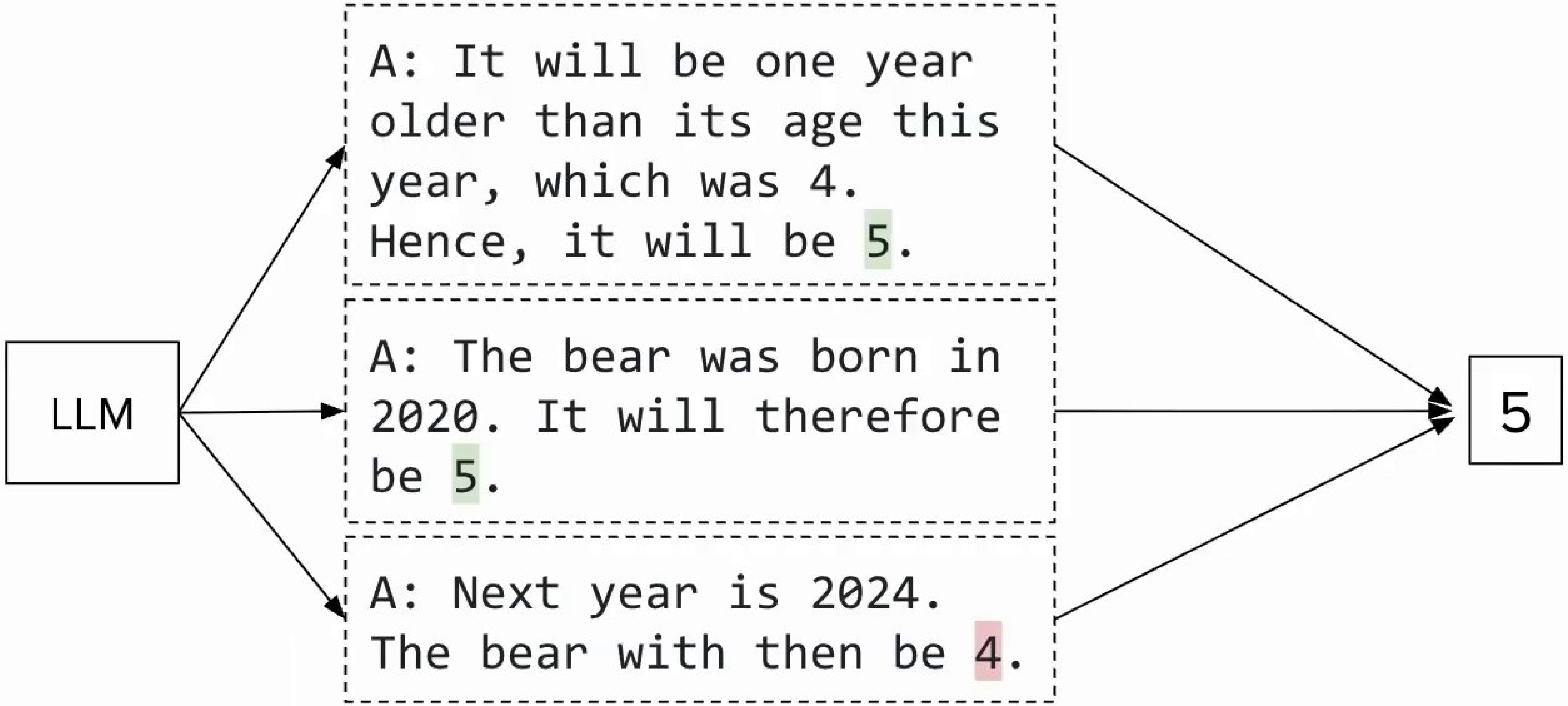
Discussion.

- Interpretability + explanation
- More tokens: higher cost and latency

Stanford

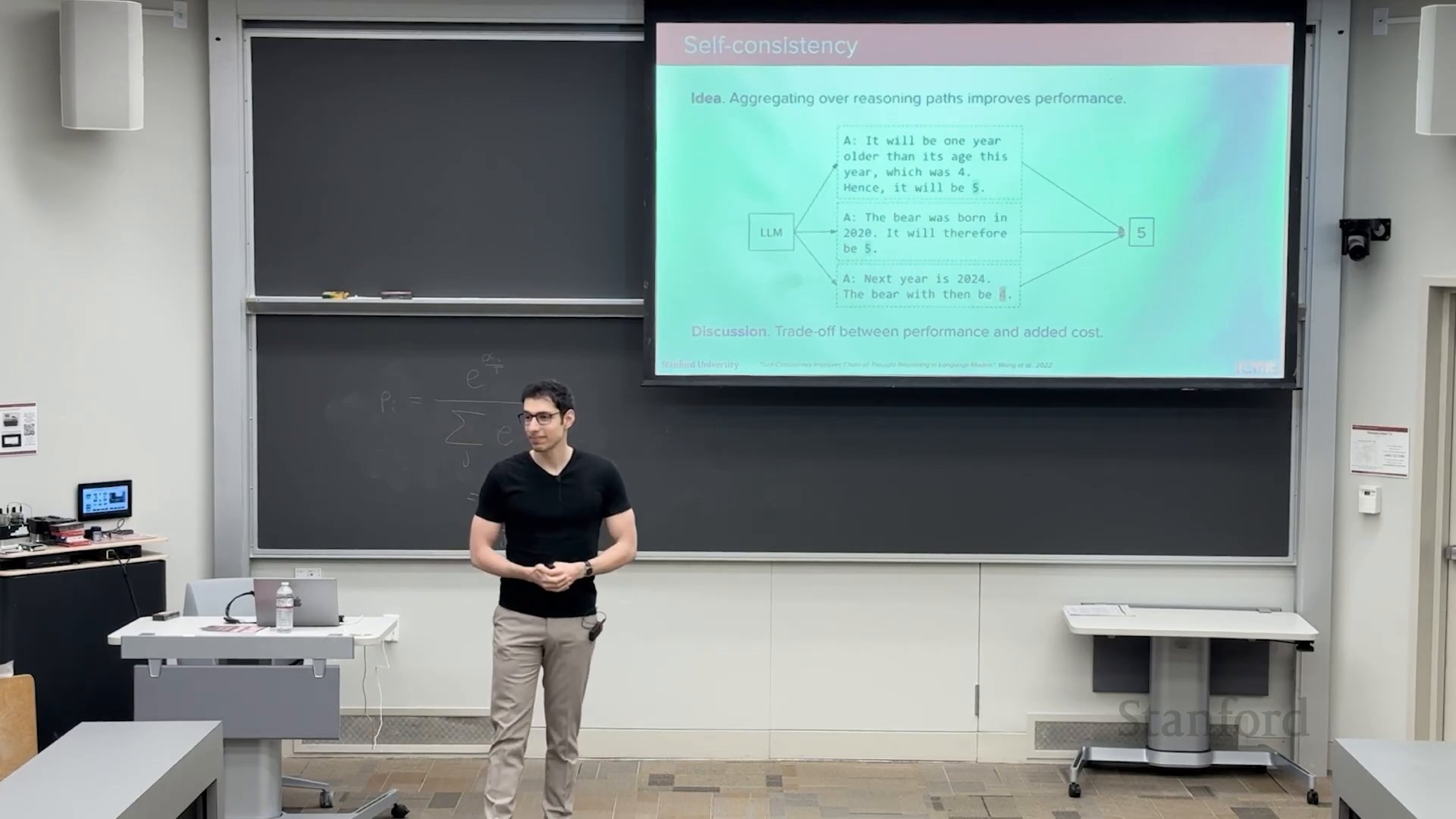
Self-consistency

Idea. Aggregating over reasoning paths improves performance.

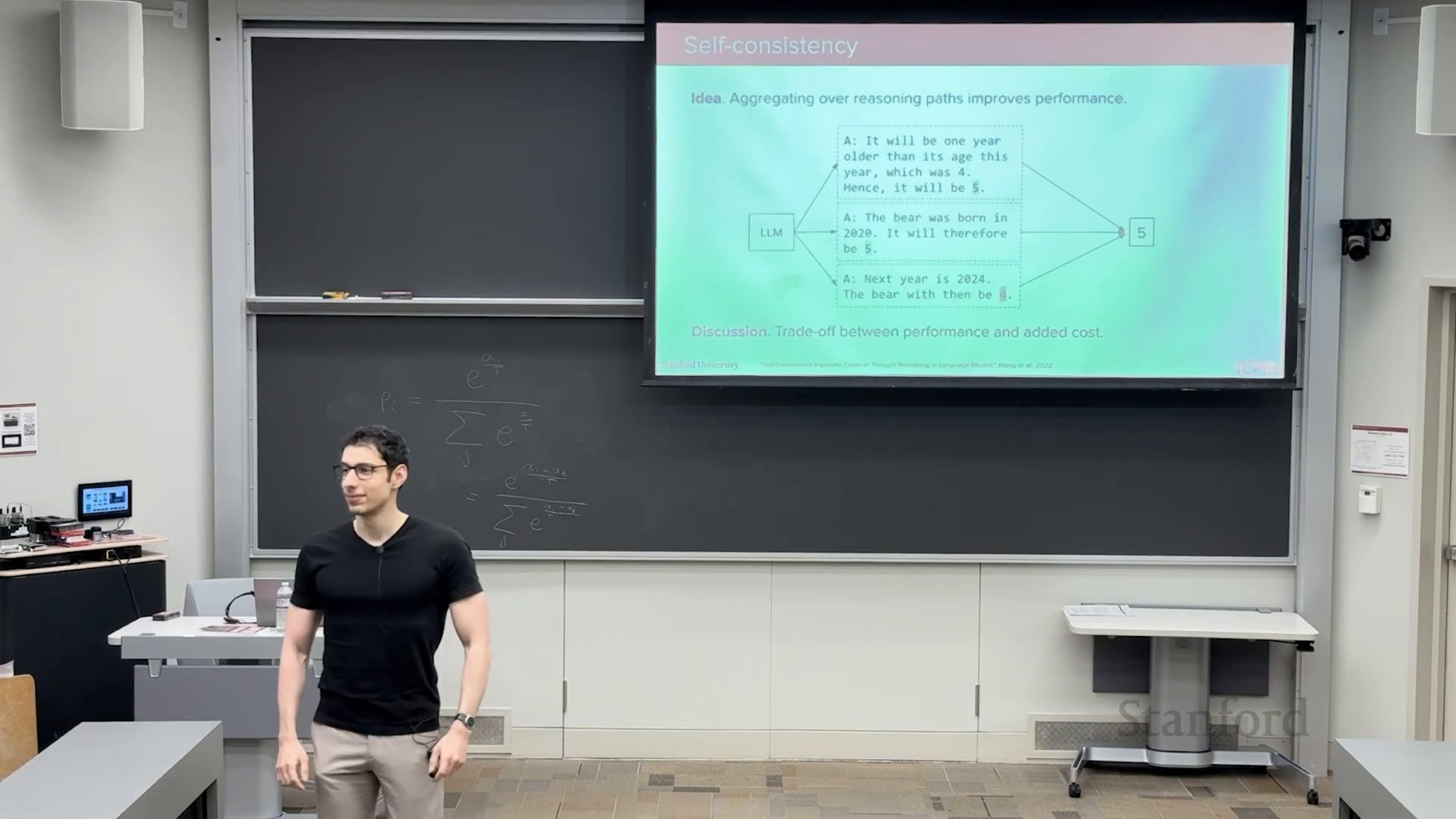


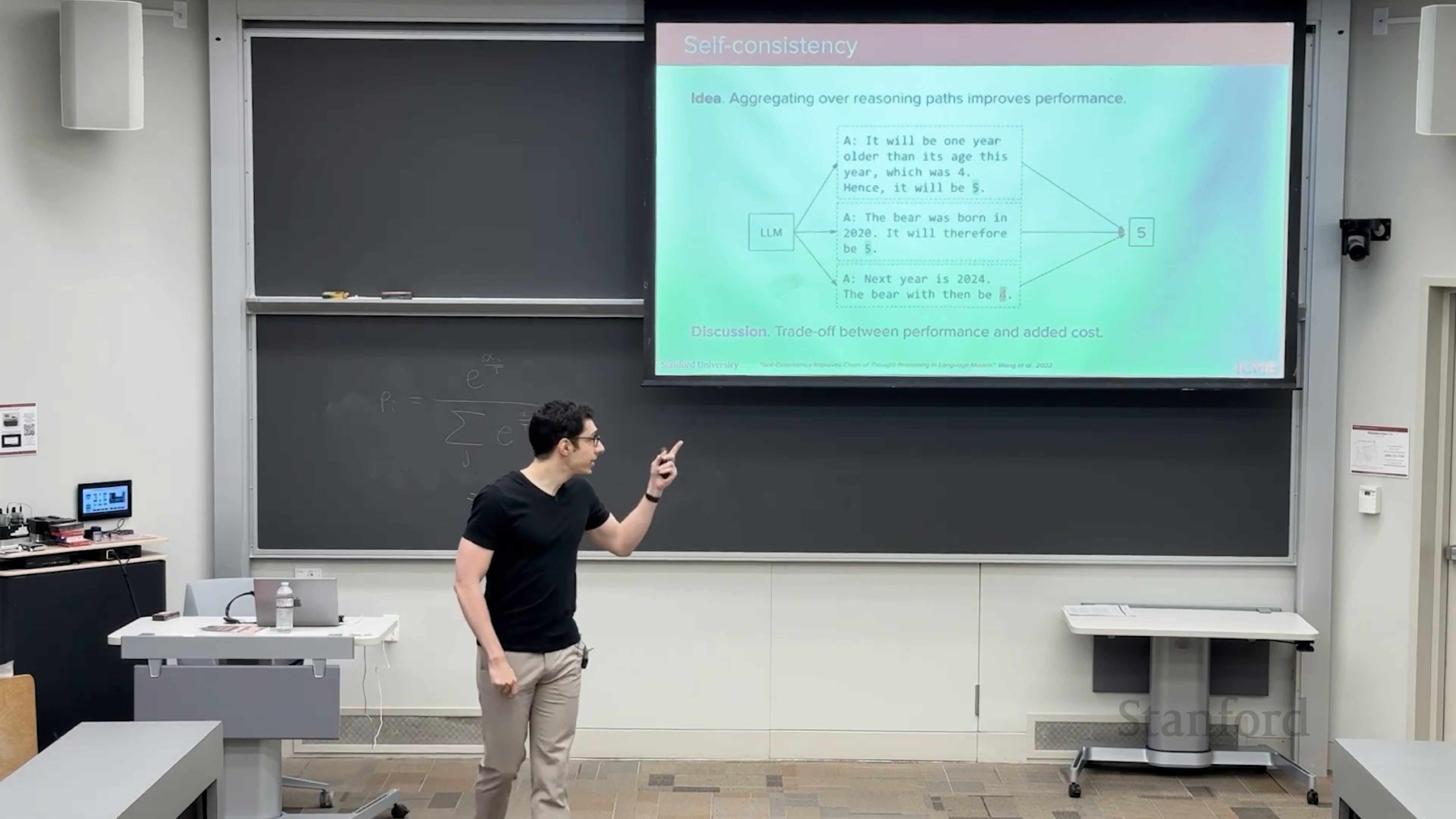
Discussion. Trade-off between performance and added cost.

Stanford



Self-consistency
Idea. Aggregating over reasoning paths improves performance.
LLM
A: It will be one year older than its age this year, which was 4. Hence, it will be 5.
A: The bear was born in 2020. It will therefore be 5.
A: Next year is 2024. The bear with then be 4.
5
Discussion. Trade-off between performance and added cost.
Stanford University
Wang et al., 2022
ICME
Stanford


Self-consistency
Idea. Aggregating over reasoning paths improves performance.
LLM
A: It will be one year older than its age this year, which was 4. Hence, it will be 5.
A: The bear was born in 2020. It will therefore be 5.
A: Next year is 2024. The bear with then be 4.
5
Discussion. Trade-off between performance and added cost.
Stanford University
"Self-Consistency Improves Chain of Thought Reasoning in Language Models" Wang et al. 2022
ICME
Stanford


Self-consistency
Idea. Aggregating over reasoning paths improves performance.
LLM
A: It will be one year older than its age this year, which was 4. Hence, it will be 5.
A: The bear was born in 2020. It will therefore be 5.
A: Next year is 2024. The bear with then be 4.
5
Discussion. Trade-off between performance and added cost.
Stanford University
"Self-Consistency Improves Chain of Thought Reasoning in Language Models" Wang et al. 2022
ICME
Stanford

Transformers & Large Language Models

LLM overview

MoE-based LLMs

Response generation

Prompting strategies

Inference optimizations

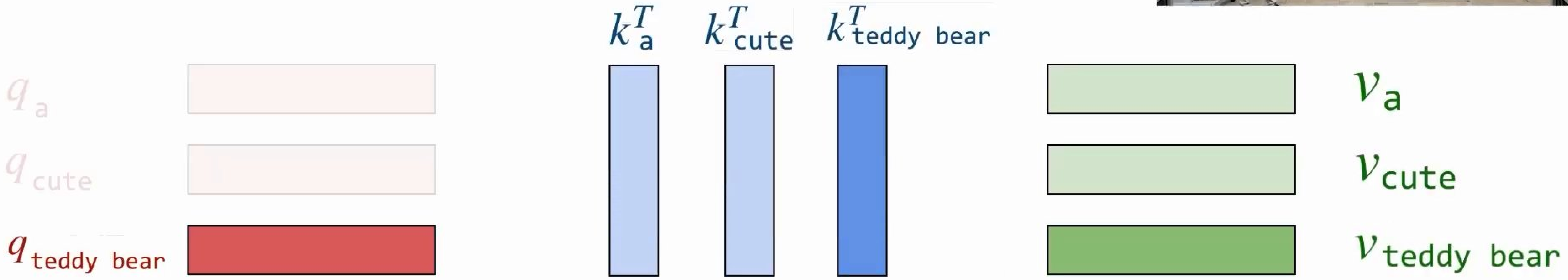
Stanford

Challenges

Motivation. Computations are expensive, any way to reduce co

Categories. Many dimensions to optimize for.

Sharing attention heads




Sharing attention heads
k_a^T
k_{cute}^T
$k_{teddy\ bear}^T$
v_a
v_{cute}
$v_{teddy\ bear}$
$q_{teddy\ bear}$
Stanford University
Stanford

Sharing attention heads

Idea. Share key/value attention heads within groups of queries

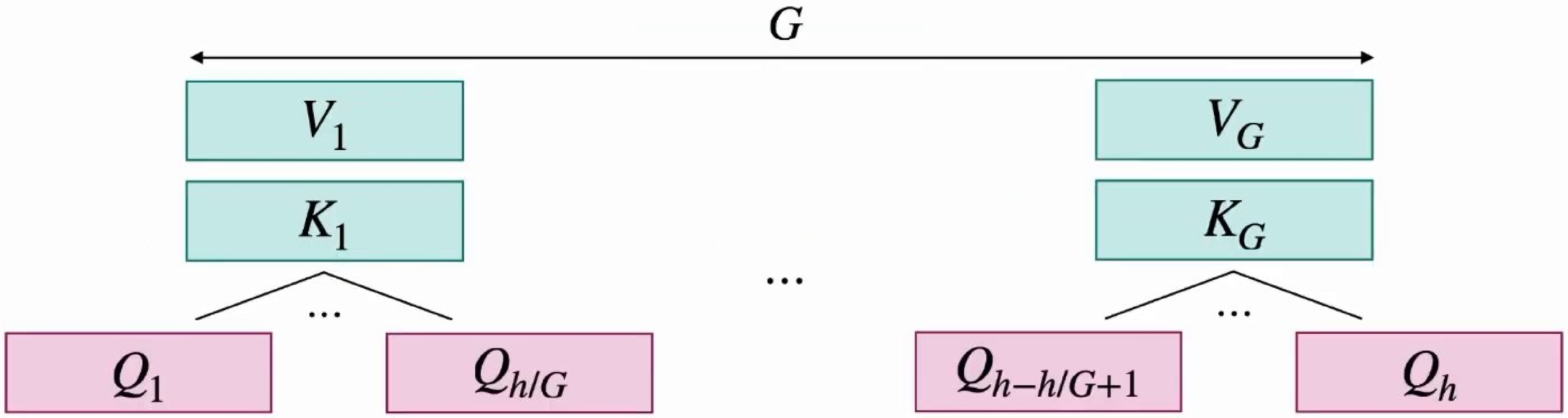


Figure from "Super Study Guide: Transformers & Large Language Models", Amidi et al., 2024.

Sharing attention heads

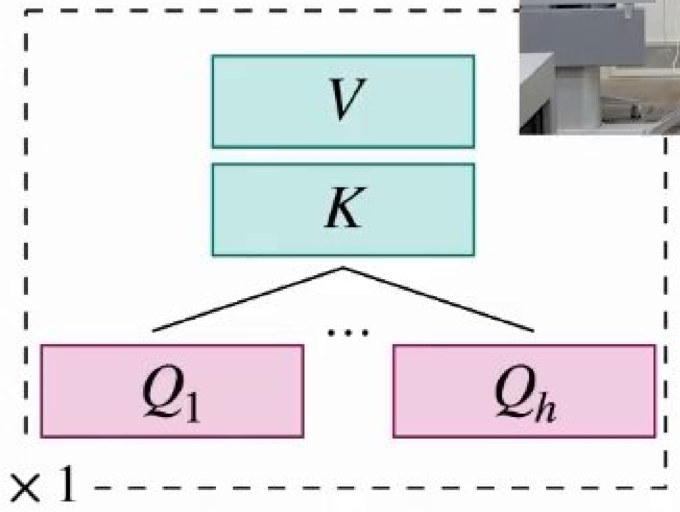
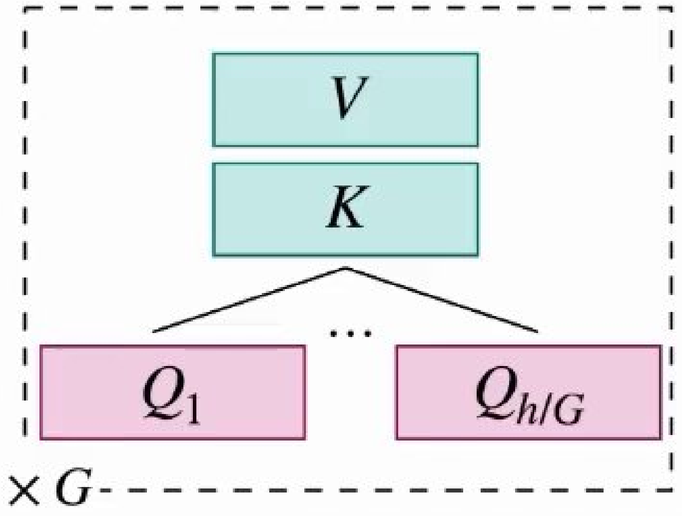
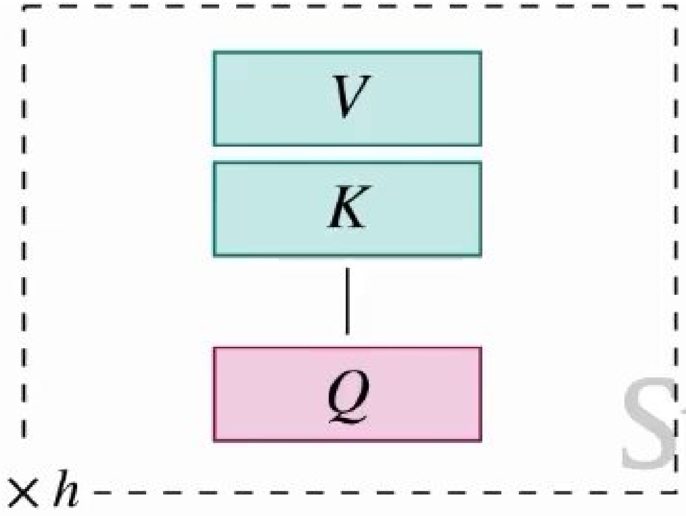
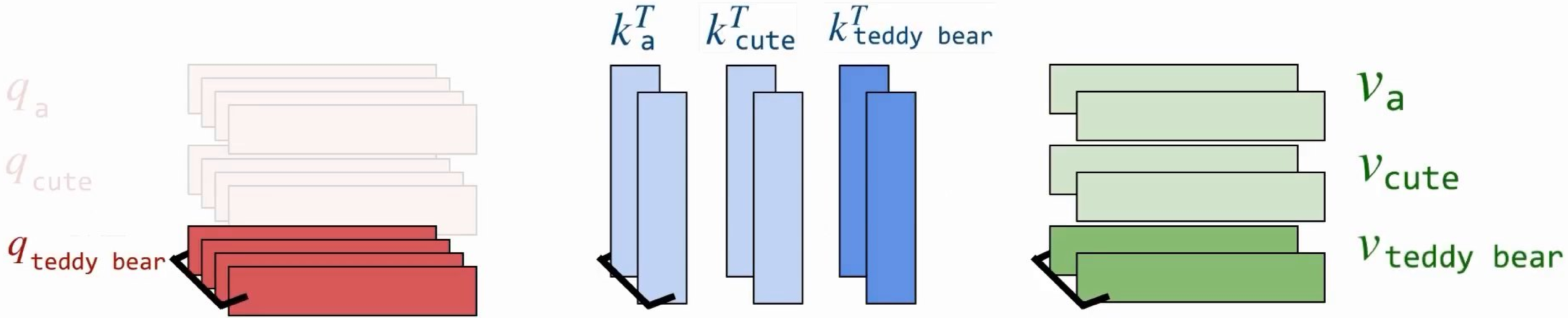
$G = 1$	**M**ulti-**Q**uery **A**ttention (MQA)	
$1 < G < h$	**G**roup-**Q**uery **A**ttention (GQA)	
$G = h$	**M**ulti-**H**ead **A**ttention (MHA)	

Figure from "Super Study Guide: Transformers & Large Language Models", Amidi et al., 2024.

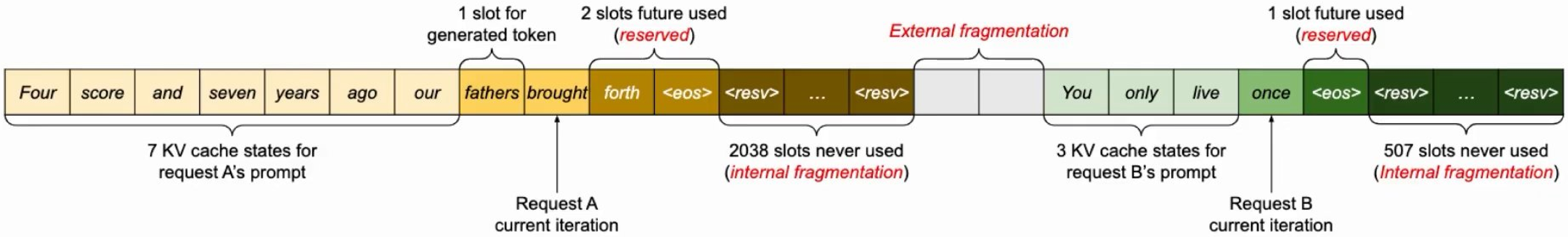
Sharing attention heads with GQA



Number of **query heads**	Number of **key heads**	Number of **value heads**
= h	= G < h	= G < h

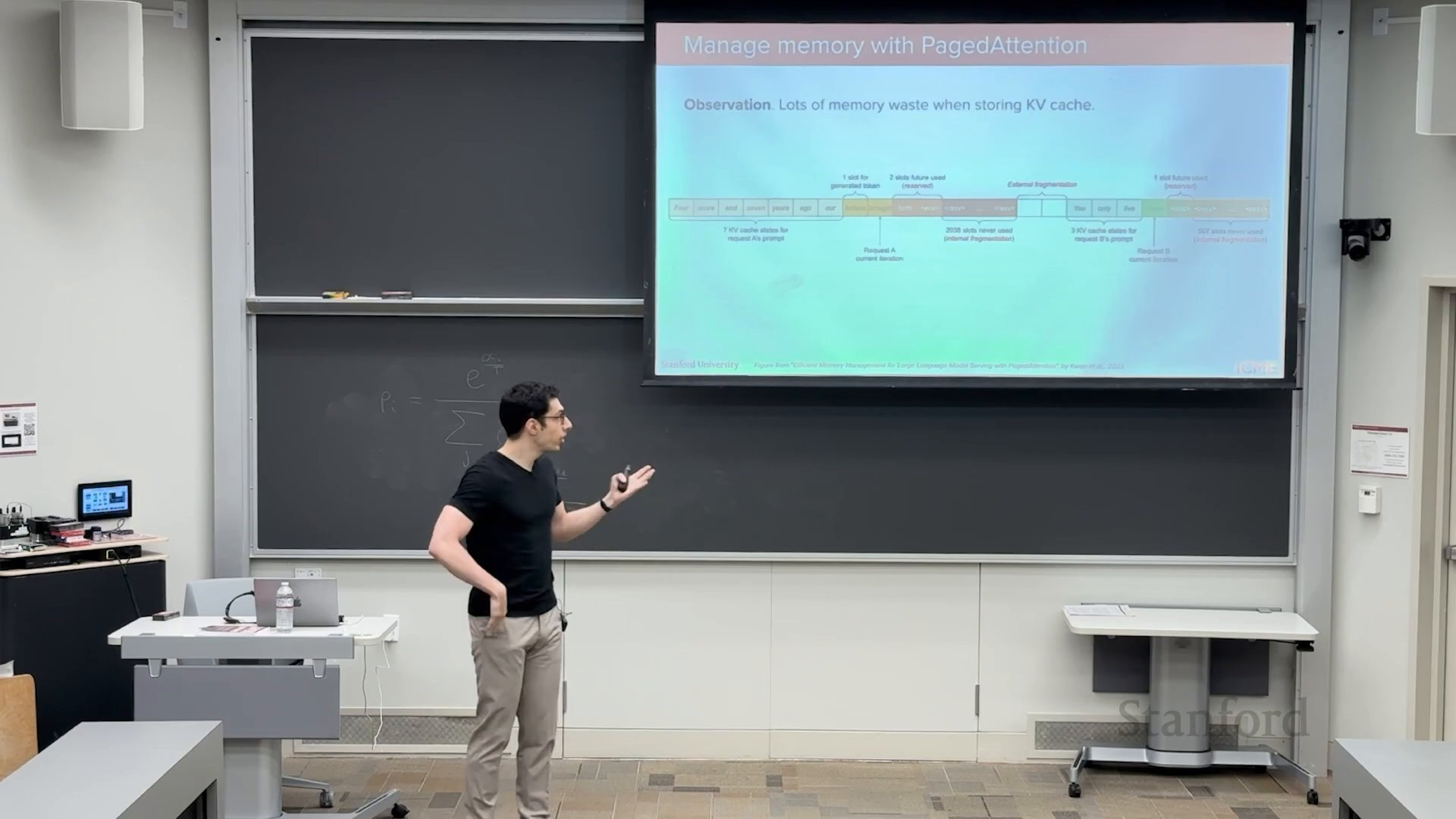
Manage memory with PagedAttention

Observation. Lots of memory waste when storing KV cache.



Stanford

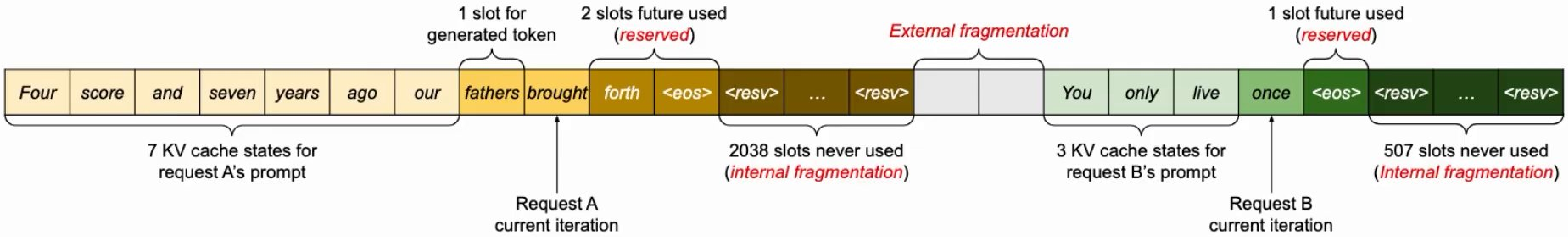
Figure from "Efficient Memory Management for Large Language Model Serving with PagedAttention", by Kwon et al., 2023.



Manage memory with PagedAttention
Observation. Lots of memory waste when storing KV cache.
1 slot for generated token
2 slots future used (reserved)
External fragmentation
1 slot future used (reserved)
Four
score
and
seven
years
ago
our
You
only
live
7 KV cache states for request A's prompt
Request A current iteration
2038 slots never used (internal fragmentation)
3 KV cache states for request B's prompt
Request B current iteration
507 slots never used (internal fragmentation)
Stanford University
Stanford

Manage memory with PagedAttention

Observation. Lots of memory waste when storing KV cache.

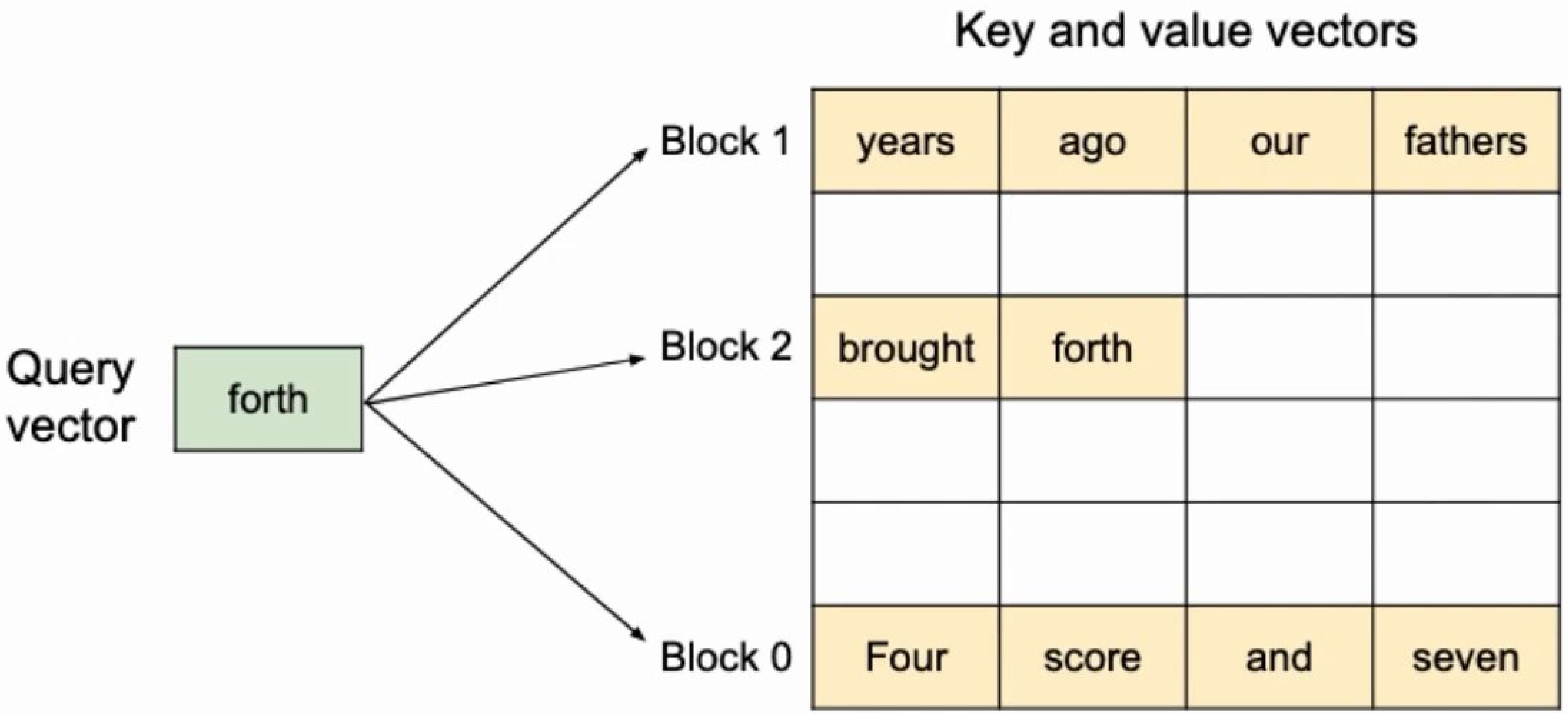


Stanford

Figure from "Efficient Memory Management for Large Language Model Serving with PagedAttention", by Kwon et al., 2023.

Manage memory with PagedAttention

Idea. Store K and V in non-contiguous space to minimize wasted memory.

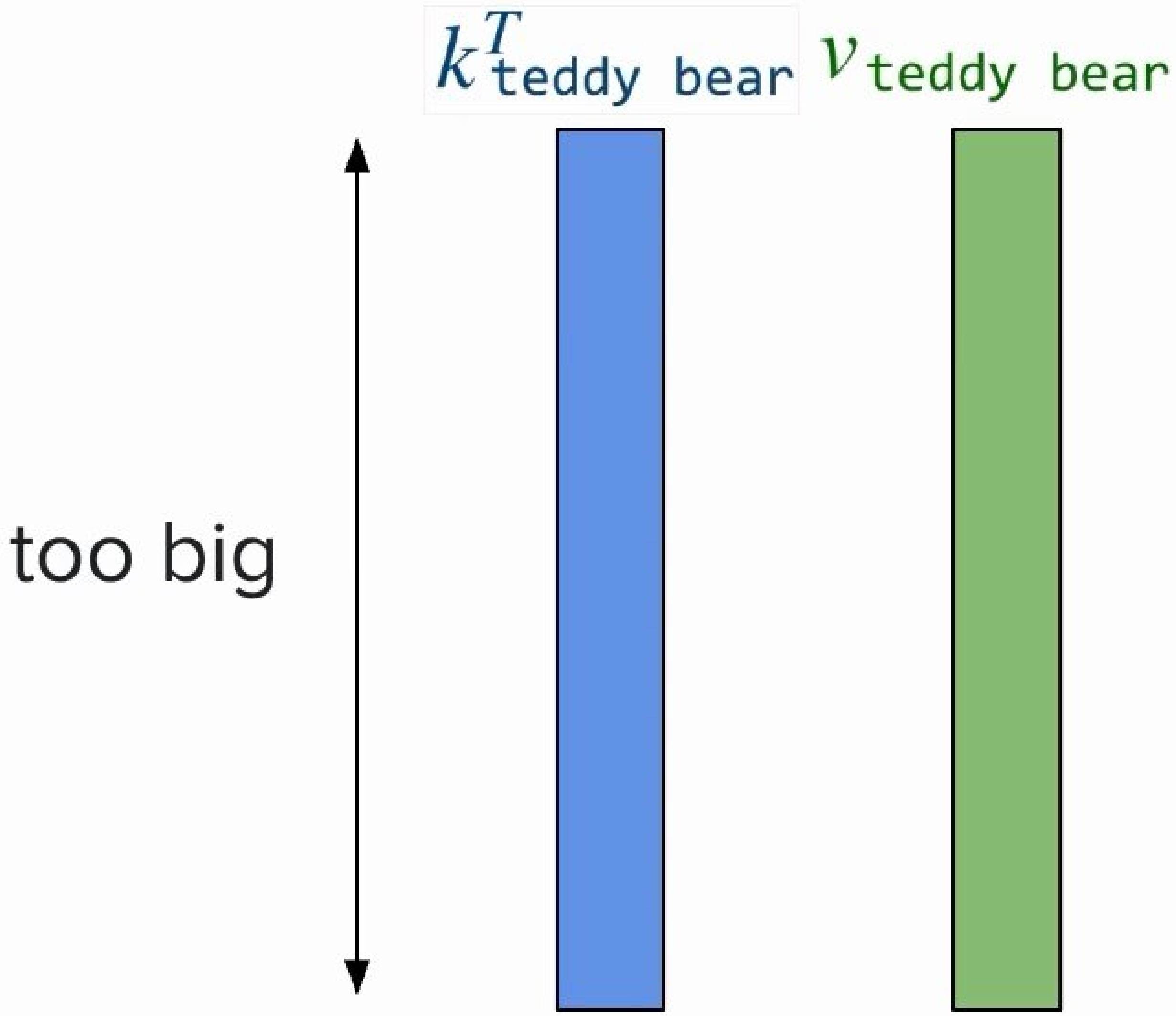


Stanford

Figure from "Efficient Memory Management for Large Language Model Serving with PagedAttention", by Kwon et al., 2023.

Reduce memory of KV cache with latent a

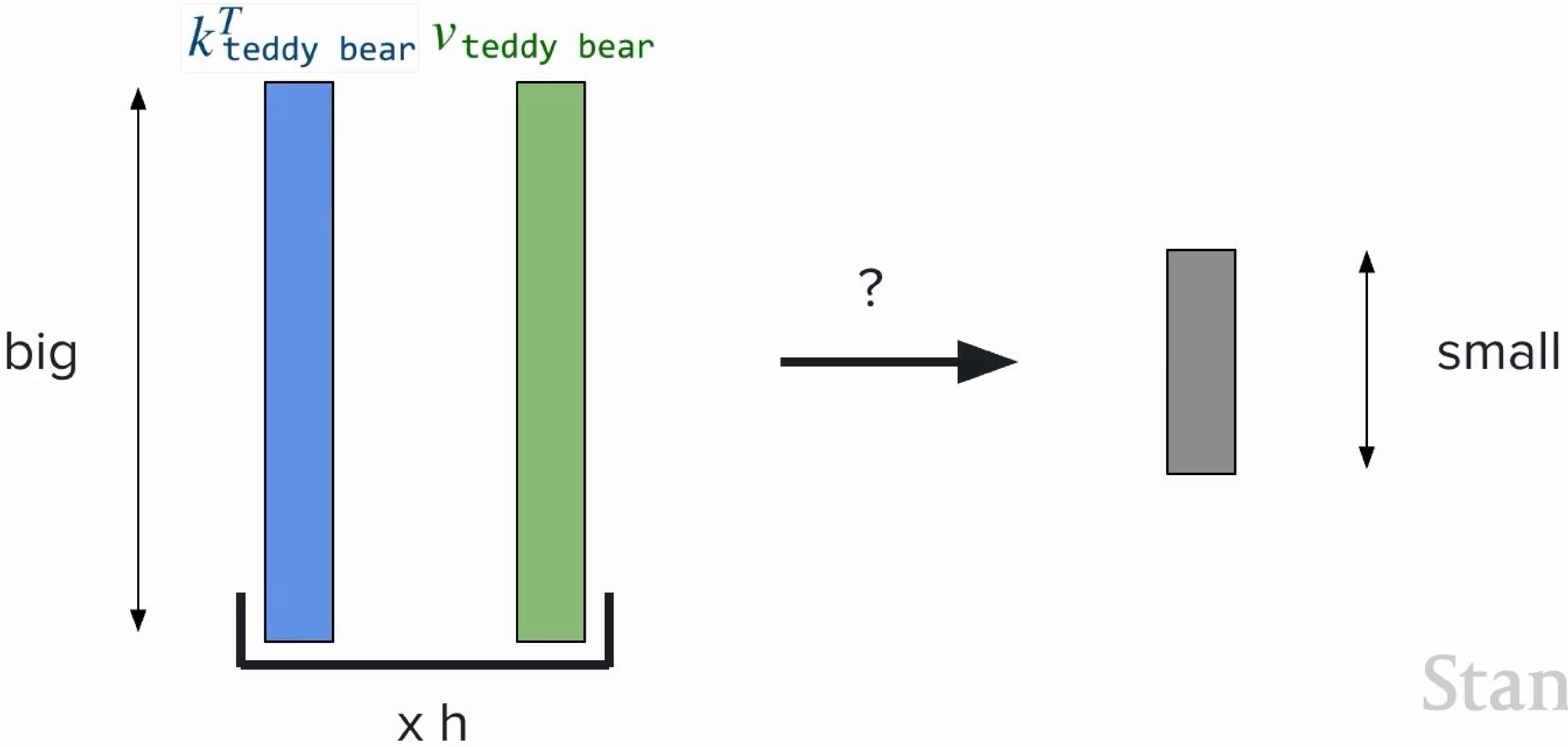
Goal. Reduce dimension of K and V stored in memory.



Stanford

Reduce memory of KV cache with latent a

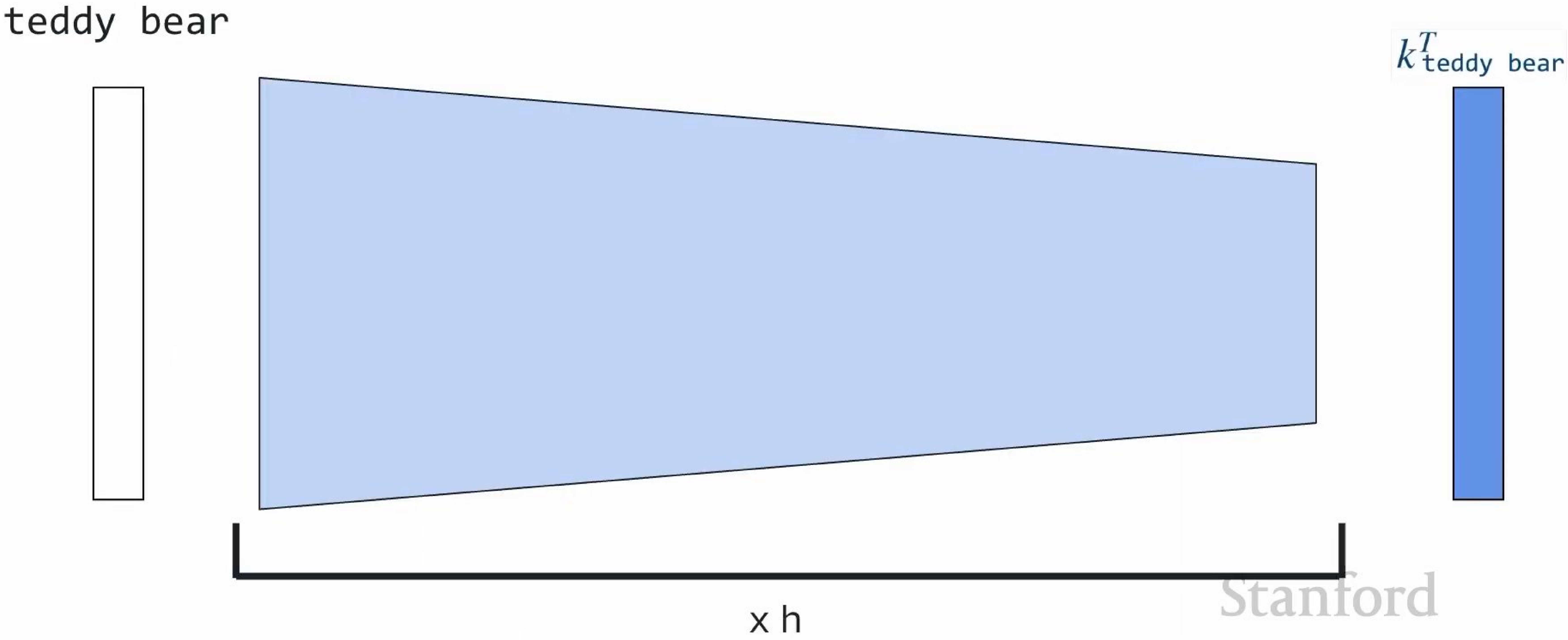
Solution. Store compressed representations instead!



"*DeepSeek-V2: A Strong, Economical, and Efficient Mixture-of-Experts Language Model*", by DeepSeek, 2023.

Reduce memory of KV cache with latent a

BEFORE



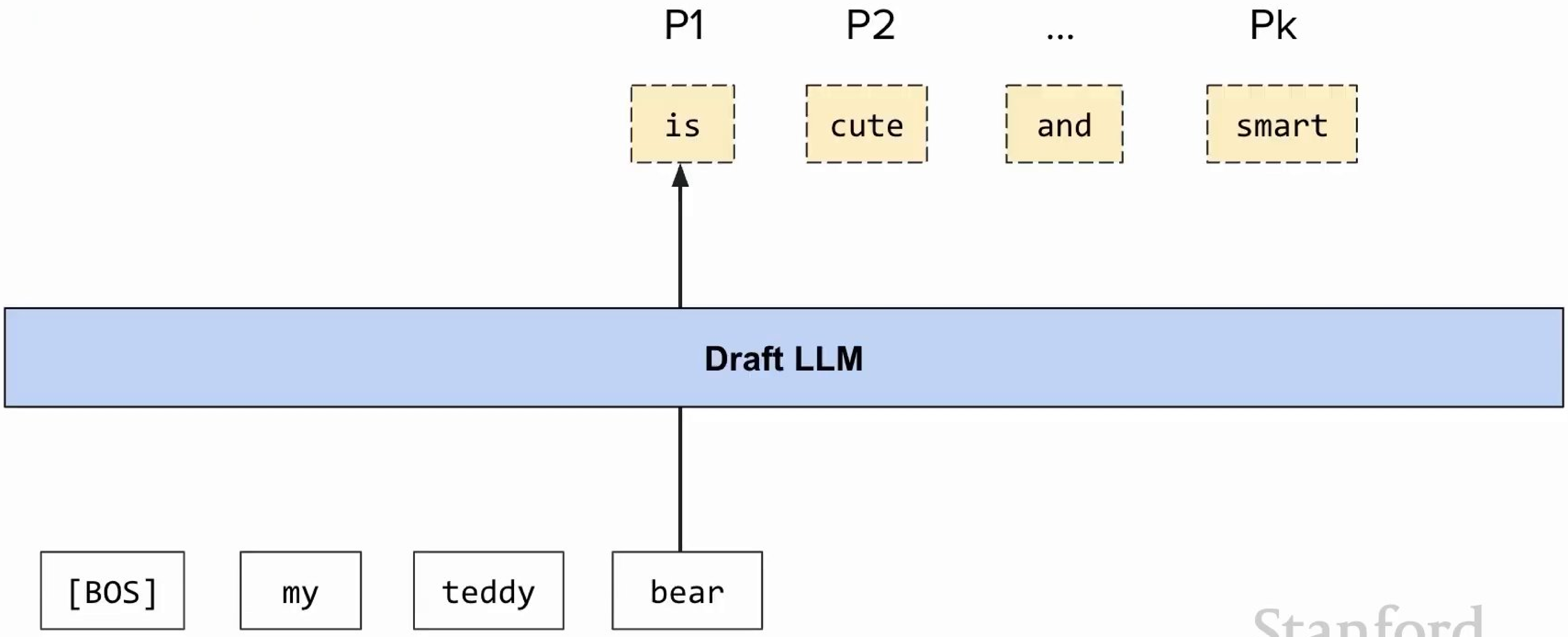
Token generation tricks
Output
Probabilities
Softmax
Linear
Stanford University
ICME
Stanford

Speeding up decoding with speculative c

Idea. Use a draft (small) model to generate tokens validated by a target (big) model.

Stanford

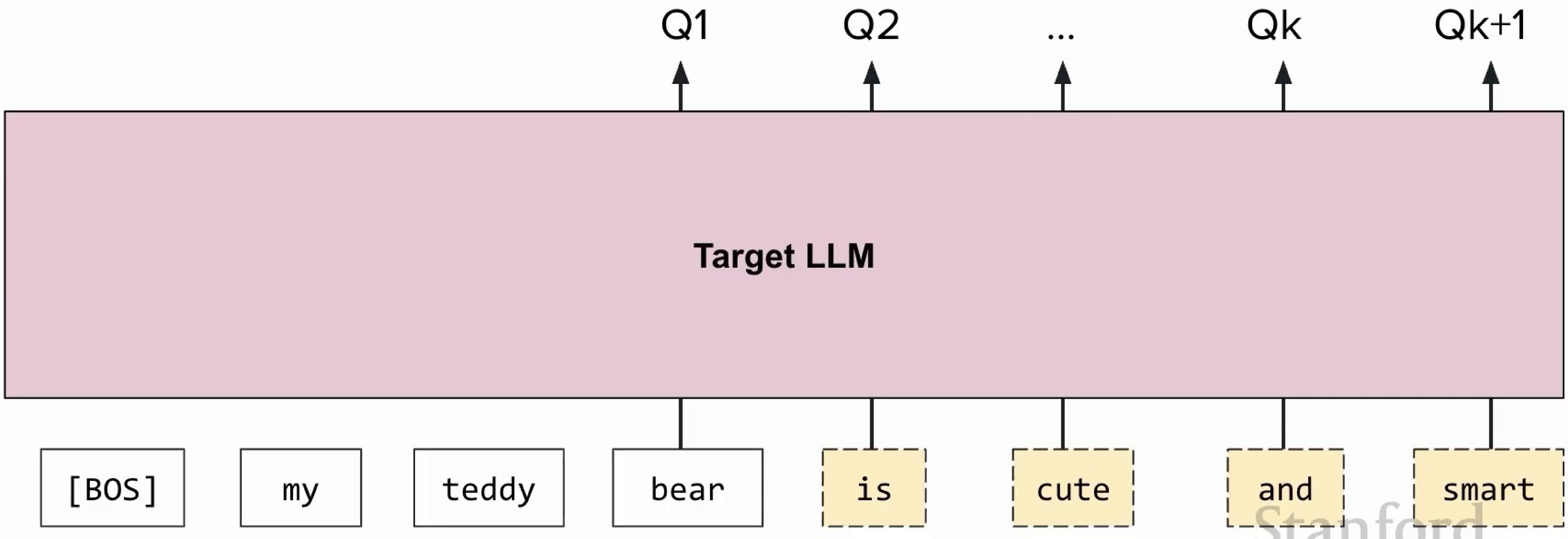
Speeding up decoding with speculative d



Stanford

"*Accelerating Large Language Model Decoding with Speculative Sampling*", Chen et al., 2023.

Speeding up decoding with speculative d



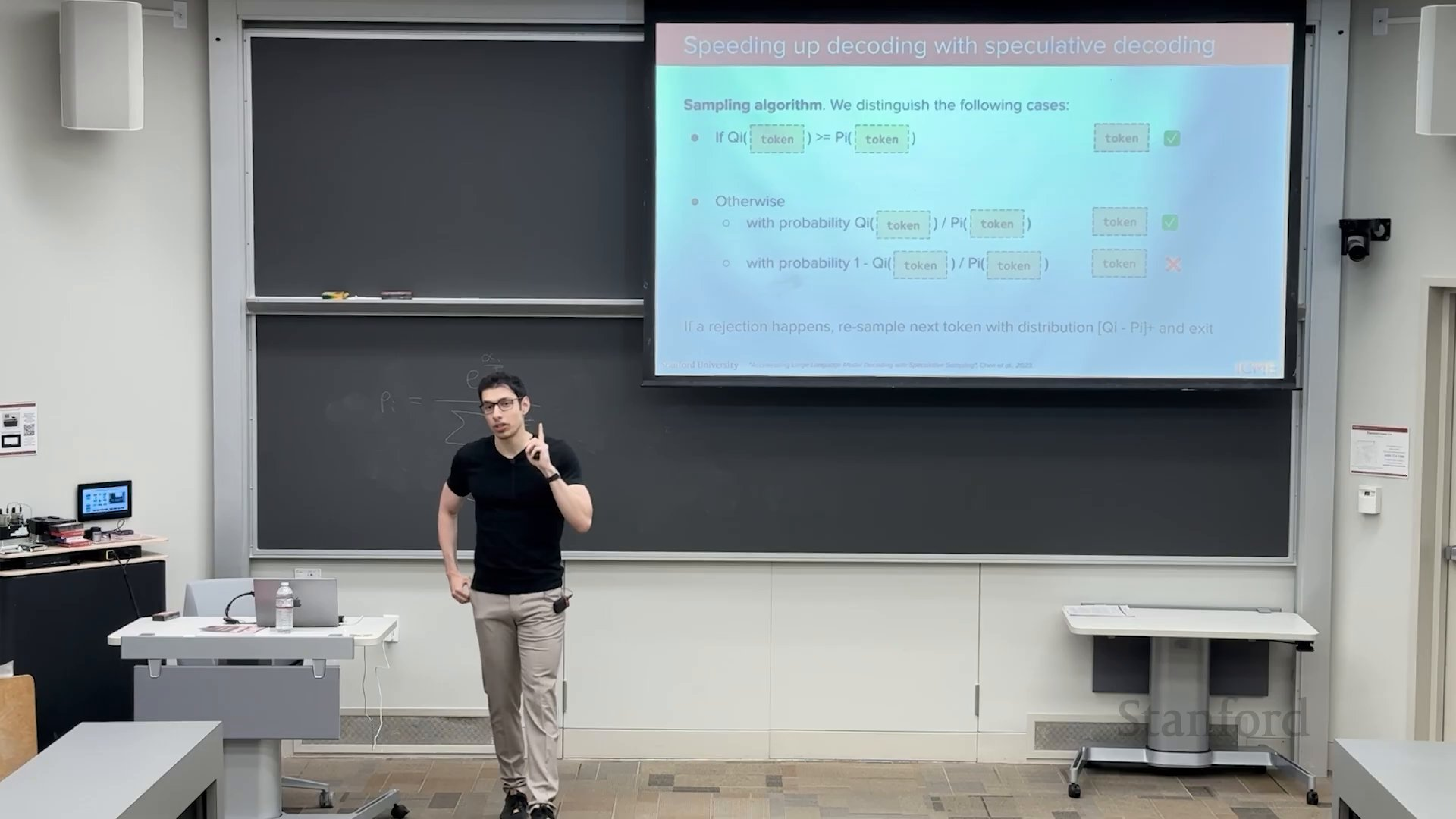
 "Accelerating Large Language Model Decoding with Speculative Sampling", Chen et al., 2023.

Speeding up decoding with speculative d

Sampling algorithm. We distinguish the following cases:

- If Qi(token) >= Pi(token) token ✅
- Otherwise
  - with probability Qi(token) / Pi(token) token ✅
  - with probability 1 - Qi(token) / Pi(token) token ❌

If a rejection happens, re-sample next token with distribution [Qi - Pi]+ and exit


Speeding up decoding with speculative decoding
Sampling algorithm. We distinguish the following cases:
If Qi(token) >= Pi(token)
token
Otherwise
with probability Qi(token) / Pi(token)
token
with probability 1 - Qi(token) / Pi(token)
token
If a rejection happens, re-sample next token with distribution [Qi - Pi]+ and exit
Stanford University
ICME
Stanford

Speeding up decoding with speculative d

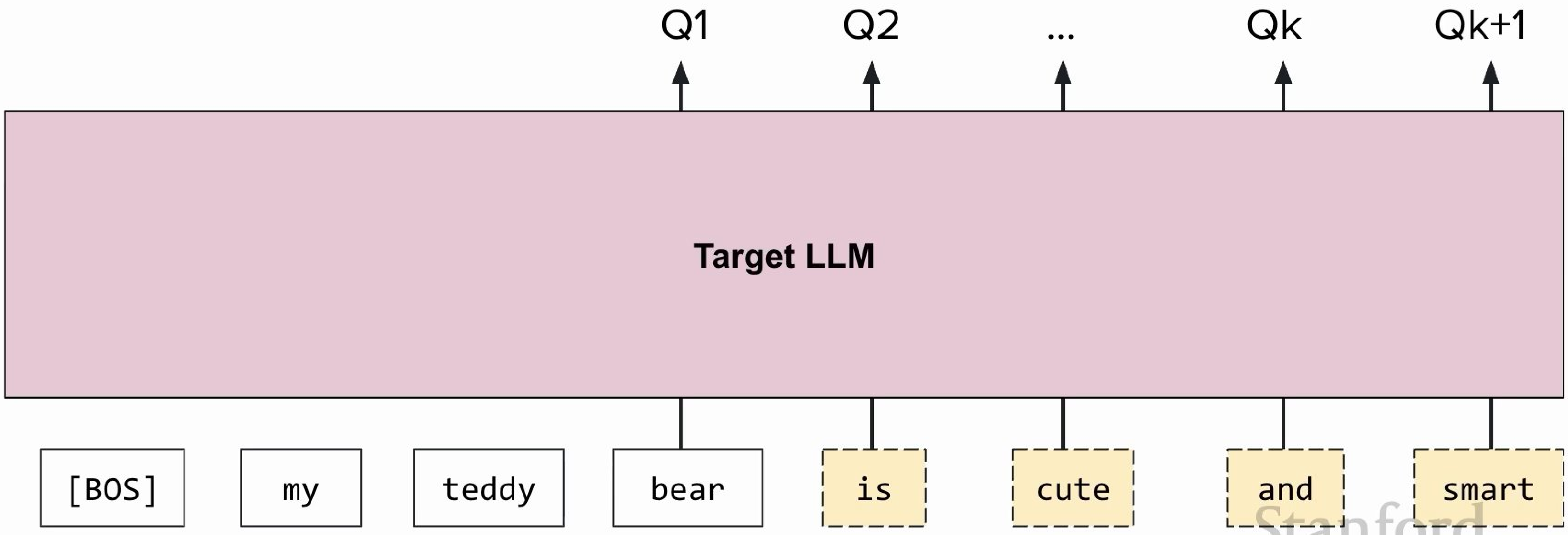
Sampling algorithm. We distinguish the following cases:

- If Qi(token) >= Pi(token) token ✅

- Otherwise
  - with probability Qi(token) / Pi(token) token ✅
  - with probability 1 - Qi(token) / Pi(token) token ❌

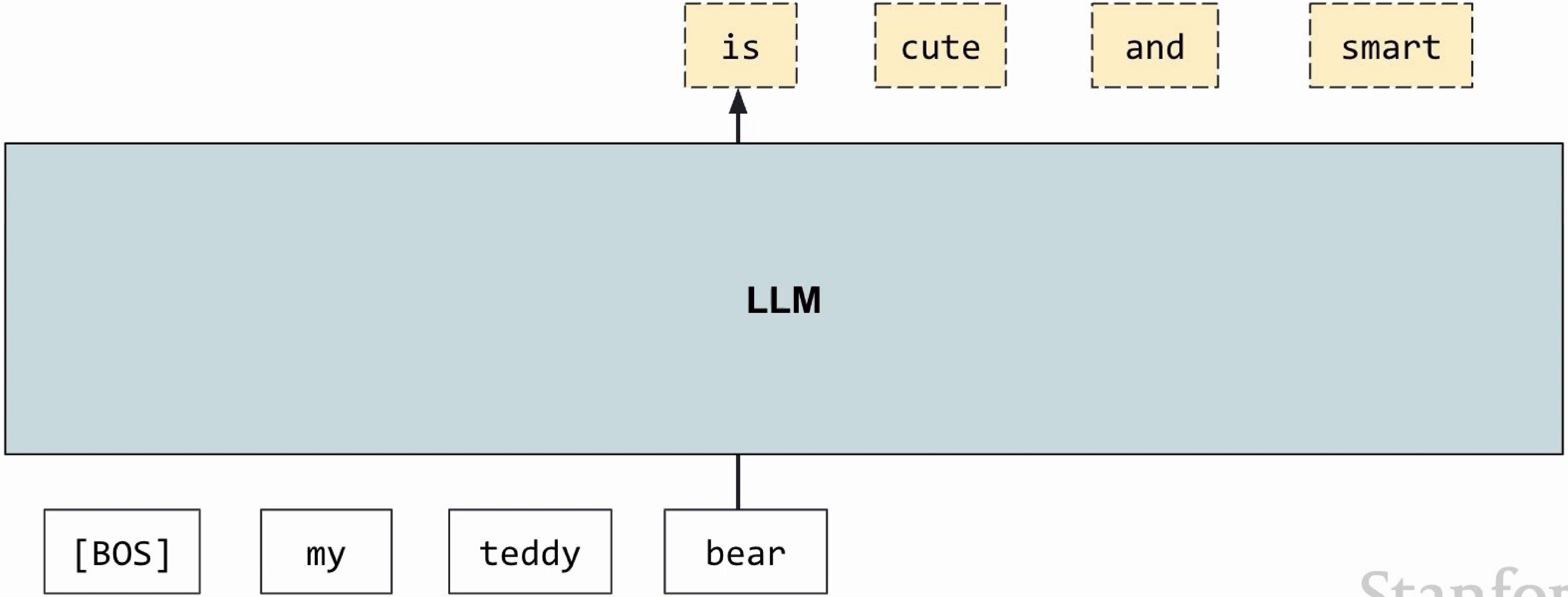
If a rejection happens, re-sample next token with distribution [Qi - Pi]+ and exit

Speeding up decoding with speculative d



Generate several tokens at once via MTP

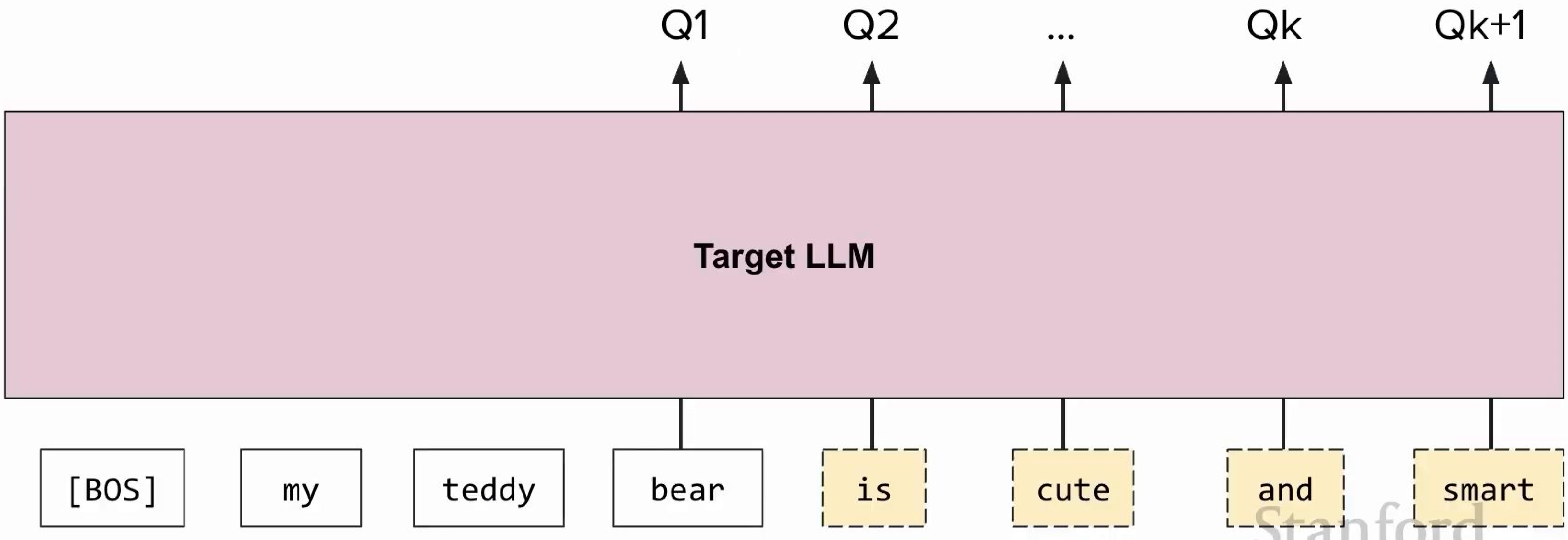
MTP = **M**ulti-**T**oken **P**rediction



Stanford

"*Better & Faster Large Language Models via Multi-token Prediction*", *Gloeckle et al., 2024.*

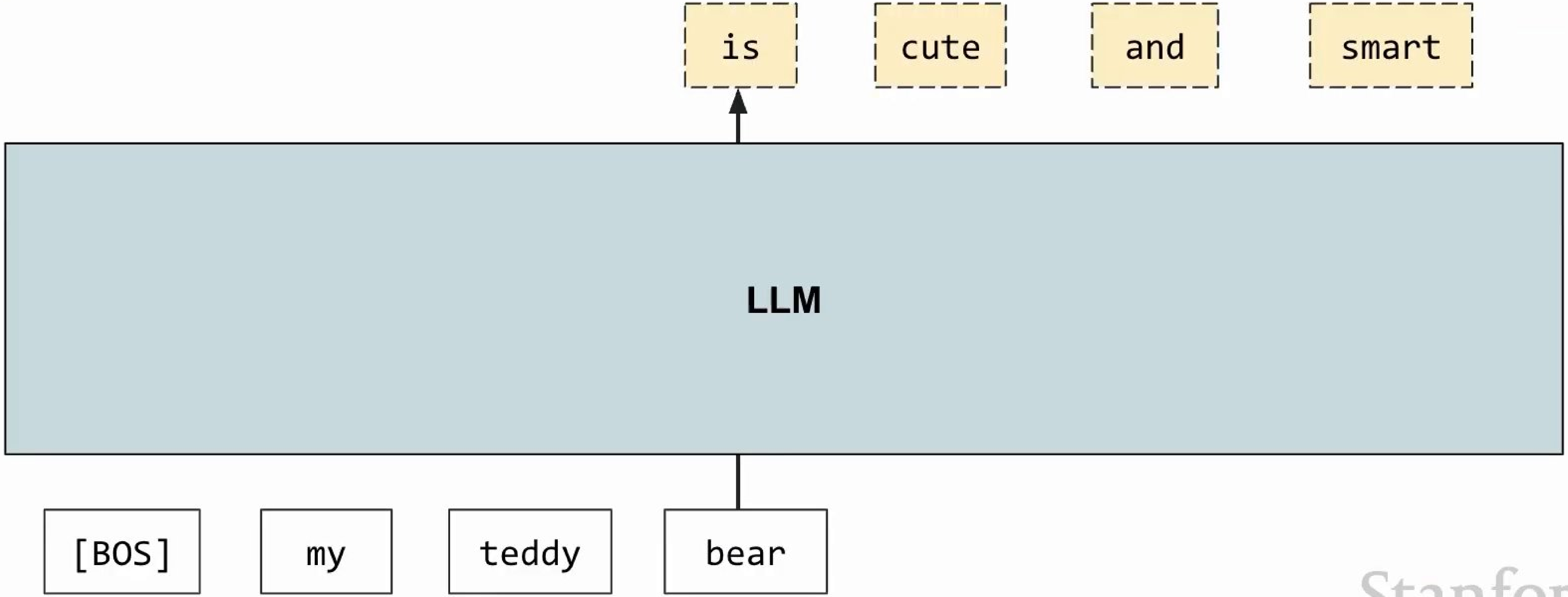
Speeding up decoding with speculative d



"*Accelerating Large Language Model Decoding with Speculative Sampling*", Chen et al., 2023.

Generate several tokens at once via MTP

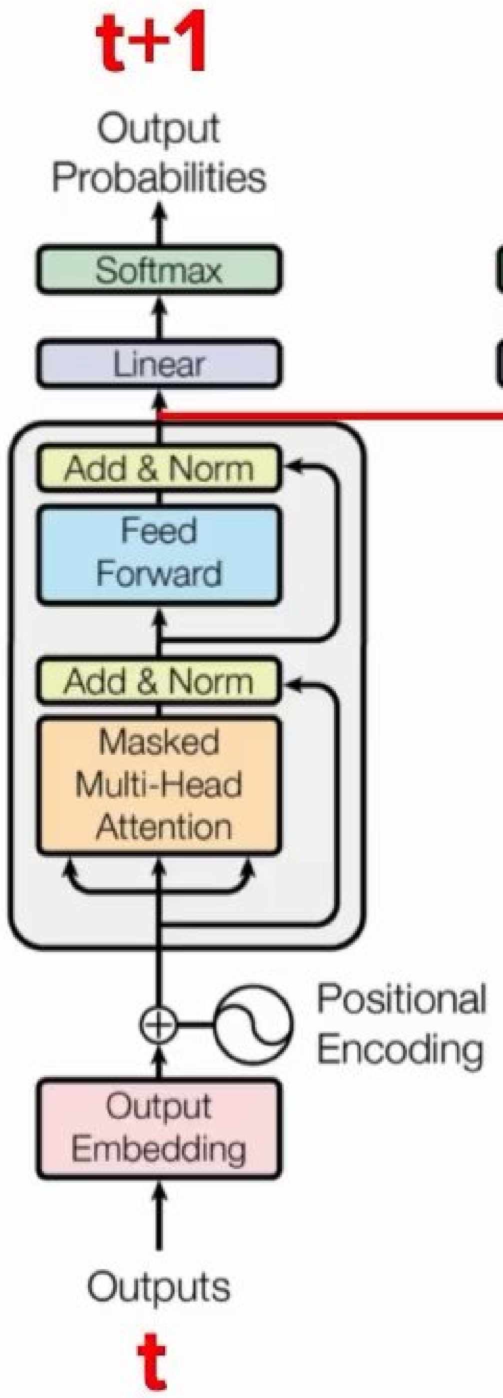
MTP = **M**ulti-**T**oken **P**rediction



Stanford

Generate several tokens at once via MTP

Idea. Train k prediction heads: same draft and target models!



Stanford

Figure adapted from "Attention is All You Need", Vaswani et al., 2017.

Challenges

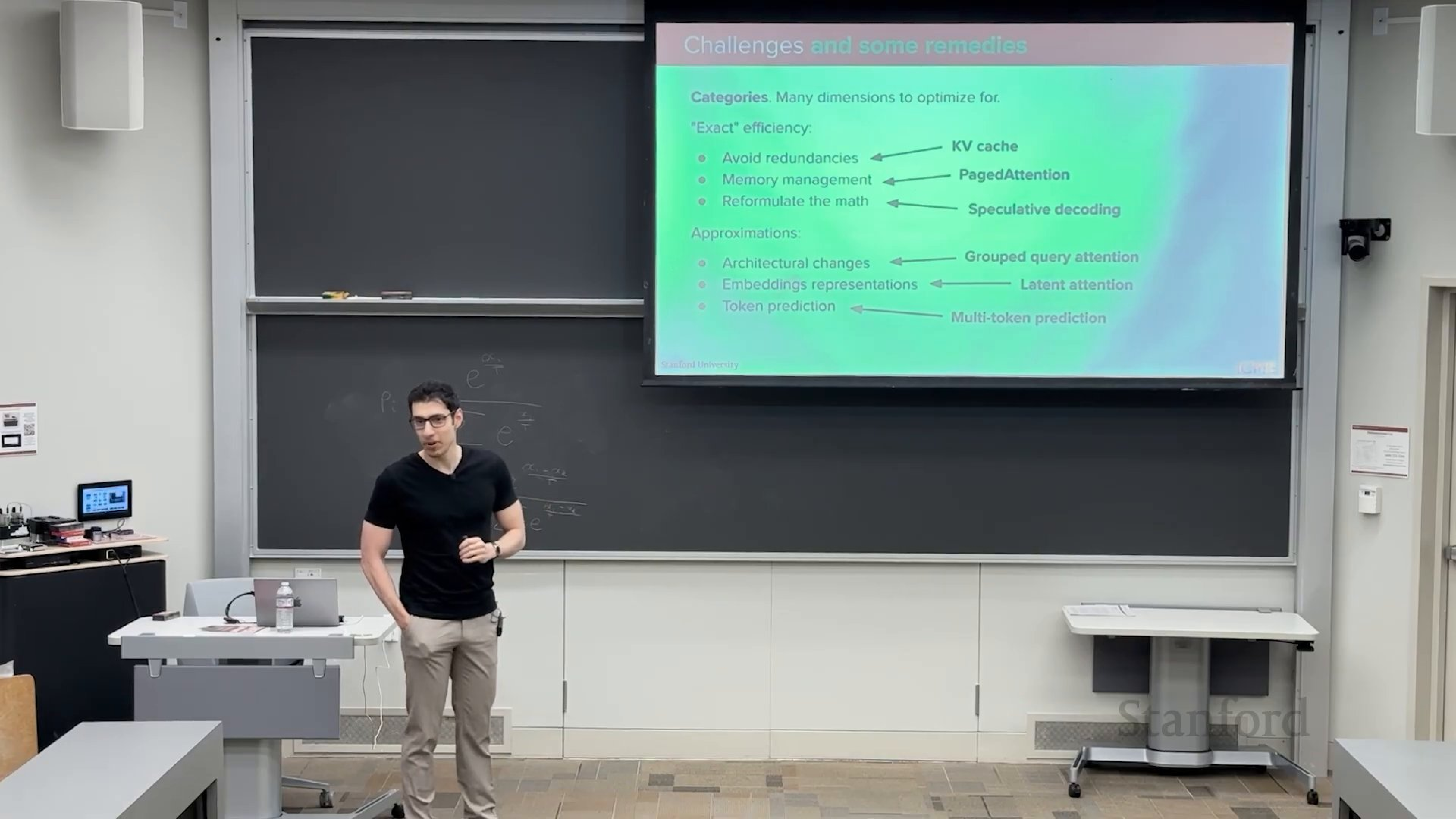
Categories. Many dimensions to optimize for.

"Exact" efficiency:

- Avoid redundancies
- Memory management
- Reformulate the math

Approximations:

- Architectural changes
- Embeddings representations
- Token prediction


Challenges and some remedies
Categories. Many dimensions to optimize for.
"Exact" efficiency:
Avoid redundancies
KV cache
Memory management
PagedAttention
Reformulate the math
Speculative decoding
Approximations:
Architectural changes
Grouped query attention
Embeddings representations
Latent attention
Token prediction
Multi-token prediction
Stanford University
Stanford

Stanford | ENGINEERING